HITACHI
Inspire the Next

Hitachi Universal Storage Platform V

Cross-OS File Exchange User's Guide

FASTFIND LINKS

Document Organization

Product Version

Getting Help

Contents

Hitachi Data Systems

MK-96RD647-01

Copyright © 2007 Hitachi Data Systems Corporation,
ALL RIGHTS RESERVED

Notice: No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or stored in a database or retrieval system for any purpose without the express written permission of Hitachi Data Systems Corporation (hereinafter referred to as “Hitachi Data Systems”).

Hitachi Data Systems reserves the right to make changes to this document at any time without notice and assumes no responsibility for its use. Hitachi Data Systems products and services can only be ordered under the terms and conditions of Hitachi Data Systems’ applicable agreements. All of the features described in this document may not be currently available. Refer to the most recent product announcement or contact your local Hitachi Data Systems sales office for information on feature and product availability.

This document contains the most current information available at the time of publication. When new and/or revised information becomes available, this entire document will be updated and distributed to all registered users.

Hitachi Data Systems is a registered trademark and service mark of Hitachi, Ltd., and the Hitachi Data Systems design mark is a trademark and service mark of Hitachi, Ltd.

All other brand or product names are or may be trademarks or service marks of and are used to identify products or services of their respective owners.

Contents

Preface vii

Overview of Hitachi Cross-OS File Exchange (FX) 1-1

About Cross-OS File Exchange Operations 2-1

Components 2-2
FX Volume Types 2-4
FCU File Transfer Options 2-7
Code Conversion (CC) Option 2-8
PIPE Function 2-11
Using the PIPE function in UNIX Systems 2-11
FAL/FCU Pipe Function Details 2-12
Pipe Function Time-Out Value 2-12
Padding (PAD) Option 2-13
Delimiter (DEL) Option 2-14
Empty File (Emp) Option 2-14
Record Description Word (RDW) Option 2-16
VSE Record (VSE) Option 2-16
FXmto Operations 2-17
FXmto with Fixed-Length Record Format 2-18
FXmto with Variable-Length Record Format 2-19
FXmto with Multiple Volume Datasets 2-23
FXotm Operations 2-24
FXotm with Fixed-Length Record Format 2-25
FXotm with Variable-Length Record Format 2-28
FXoto Operations 2-29
Host Access and I/O Contention 2-30
AIX Shared Open Function 2-32
AIX Reserve Function 2-34
AIX Reserved Retry Function 2-34

Environment Variables and Functions 2-34
Bidirectional Data Transfer 2-36

Preparing for Cross-OS File Exchange Operations 3-1

System Requirements 3-2
VSE Requirements and Restrictions 3-4
Compiler Requirements 3-6
Maximum Data Size 3-7
Interoperability with HDLM 3-9
Installing and Configuring the FX Volumes 3-10
Installing the FAL/FCU Software 3-15
Installing FAL/FCU on UNIX-Based Platforms 3-15
32-Bit FAL/FCU Software 3-15
64-Bit FAL/FCU Software 3-18
Installing FAL/FCU on Windows 3-19
Uninstalling the FAL/FCU Software on UNIX-Based Plaforms 3-20
Uninstalling FAL/FCU on Windows 3-20
Entering the FX License Key Code 3-21
Creating FXoto Volumes Using the FMT Utility 3-22
Creating the FX Volume Definition File(s) 3-28
Verifying Mainframe Dataset Requirements 3-32
Allocating FXoto Intermediate Datasets 3-33

Using the Cross-OS File Exchange Software 4-1

FCU for UNIX 4-2
FCU Version and Copyright Screen 4-2
File Conversion Utility Screen 4-4
File Menu Commands 4-8
Help Menu Commands 4-8
Error Information Screen 4-10
FCU for Windows 4-10
FCU Version and Copyright Dialog 4-10
File Conversion Utility Window 4-12
Volume Information Dialog 4-14
Mainframe File Information Dialog 4-14
Option Dialog 4-16
Parameter Line Dialog 4-18
Execute Dialogs 4-18
Error Information Dialog 4-20
Log Files 4-21
Format Utility for Windows 4-22
Allocation Utility for Windows 4-27

Performing Cross-OS File Exchange Operations 5-1
Performing File Transfer Operations - UNIX 5-3
Starting the FCU GUI for UNIX 5-3
Performing File Transfer Operations (UNIX) 5-6
Using the `listvol` Function (UNIX) 5-8
Creating FCU Parameter Definition Files (UNIX) 5-9
Creating Multiple Volume Definition Files (UNIX) 5-10
Using FCU from the Command Line (UNIX) 5-12
Performing File Transfer Operations – Windows 5-17
Starting the FCU GUI 5-17
Performing File Transfer Operations (Windows) 5-19
Creating FCU Parameter Definition Files (Windows) 5-23
Creating Multiple Volume Definition Files (Windows) 5-25
Using FCU from the Command Line (Windows) 5-27

Performing File Access Library (FAL) Operations 6-1
FAL Requirements 6-2
FAL Functions 6-3
Converting Dataset Attribute Information 6-3
Opening a Dataset 6-4
Reading Data 6-5
Writing Data 6-6
Closing a Dataset 6-7
Acquiring Error Information 6-7
Acquiring Dataset Attributes 6-8
Acquiring Attribute Information for a Specific Dataset 6-8
Acquiring Attribute Information for Multiple Datasets 6-9
Converting DO and RF Information 6-12
Converting the Dataset Organization (DO) Type Value 6-12
Converting the Record Format (RF) Type Value 6-12
Using the FAL Functions 6-14
Multi-Thread Function 6-18
Information Storage Area 6-19
Open Dataset 6-20
Read Data 6-22
Write Data 6-23
Close Dataset 6-24
Free Information Stored Area 6-25
Initialize Target Record Pointer 6-26
Get Dataset Attribute Information 6-27
Get Specified Dataset Attribute Information 6-27
Get Multiple Dataset Attribute Information 6-28
Compiling 6-31
Error Information 6-32

FAL Usage Scenario 6-33

Troubleshooting 7-1

Troubleshooting 7-2
Error Codes and Messages 7-3
FCU Error Codes for UNIX 7-7
FCU Error Codes for Windows Systems 7-16
Calling the Support Center 7-21

EBCDIC-ASCII Code Conversion A-1

Acronyms and Abbreviations Acronyms-1

Index Index-1

Preface

This *Hitachi TagmaStore® USP and NSC Cross-OS File Exchange (FX) User's Guide* describes and provides instructions for installing and using the FX feature of the Hitachi TagmaStore Universal Storage Platform V (USP V) and Network Storage Controller (NSC) disk array subsystems.

Please read this document carefully to understand how to use this product, and maintain a copy for reference purposes.

This preface includes the following information:

- ☐ Intended Audience
- ☐ Product Version
- ☐ Document Revision Level
- ☐ Changes in this Revision
- ☐ Document Organization
- ☐ Referenced Documents
- ☐ Document Conventions
- ☐ Convention for Storage Capacity Values
- ☐ Getting Help
- ☐ Comments

Notice: The use of Cross-OS File Exchange and all other Hitachi Data Systems products is governed by the terms of your agreement(s) with Hitachi Data Systems.

Intended Audience

This document is intended for system administrators, Hitachi Data Systems representatives, and Authorized Service Providers who are involved in installing, configuring, and operating the Hitachi Universal Storage Platform V storage system.

This document assumes the following:

- The user has a background in data processing and understands RAID storage systems and their basic functions.
- The user is familiar with the Hitachi Universal Storage Platform V storage system and has read the *Universal Storage Platform V User and Reference Guide*.
- The user is familiar with the Storage Navigator software for the Universal Storage Platform V and has read the *Storage Navigator User's Guide*.
- The user is familiar with the operating system and web browser software on the system hosting the Storage Navigator software.

Product Version

This document revision applies to Universal Storage Platform V microcode 60-03-*x* and higher.

Document Revision Level

Revision	Date	Description
MK-96RD647-P	February 2007	Preliminary Release
MK-96RD647-00	April 2007	Initial Release, supersedes and replaces MK-96RD647-P
MK-96RD647-01	May 2007	Revision 1, supersedes and replaces MK-96RD647-00

Changes in this Revision

Not applicable to this release.

Document Organization

The following table provides an overview of the contents and organization of this document. Click the chapter title in the left column to go to that chapter. The first page of each chapter provides links to the sections in that chapter.

Chapter	Description
Overview of Cross-OS File Exchange	This chapter provides a brief overview FX functions, features, and benefits.
About FX Operations	This chapter describes how FX operates in typical system configurations and describes the operations a user may perform using FX.
Preparing for FX Operations	This chapter describes the system requirements and volume configuration necessary to run and operate FX. It also provides instructions on how to install FX.
Using FX Software	This chapter describes the user interface and the commands available to an FX user.
Performing FX Operations	This chapter describes how to operate FX.
Performing FAL Operations	This chapter describes how to perform FAL operations using FX.
Troubleshooting	This chapter provides descriptions of error codes and support desk contact data.
Acronyms and Abbreviations	Defines the acronyms and abbreviations used in this document.
Index	Lists the topics in this document in alphabetical order.

Referenced Documents

Hitachi Universal Storage Platform V:

- *Command Control Interface (CCI) User and Reference Guide*, MK-90RD011
- *LUN Manager User's Guide*, MK-96RD615
- *User and Reference Guide*, MK-96RD635
- *Storage Navigator User's Guide*, MK-96RD621

<Other Hitachi product (e.g., TagmaStore USP/NSC, HiCommand)>:

- *HiCommand® Device Manager Web Client User's Guide*, MK-91HC001

Document Conventions

The terms “Universal Storage Platform V” and “USP V” refer to all models of the Hitachi Universal Storage Platform V, unless otherwise noted.

This document uses the following typographic conventions:

Typographic Convention	Description
Bold	Indicates text on a window, other than the window title, including menus, menu options, buttons, fields, and labels. Example: Click **OK**.
Italic	Indicates a variable, which is a placeholder for actual text provided by the user or system. Example: copy *source-file target-file* ***Note:*** Angled brackets (< >) are also used to indicate variables.
`screen/code`	Indicates text that is displayed on screen or entered by the user. Example: `# pairdisplay -g oradb`
< > angled brackets	Indicates a variable, which is a placeholder for actual text provided by the user or system. Example: `# pairdisplay -g <group>` ***Note:*** Italic font is also used to indicate variables.
[ ] square brackets	Indicates optional values. Example: [a | b] indicates that you can choose a, b, or nothing.
{ } braces	Indicates required or expected values. Example: { a | b } indicates that you must choose either a or b.
| vertical bar	Indicates that you have a choice between two or more options or arguments. Examples: [a | b] indicates that you can choose a, b, or nothing. { a | b } indicates that you must choose either a or b.
underline	Indicates the default value. Example: [a | b]

This document uses the following icons to draw attention to information:

Icon	Meaning	Description
	Note	Calls attention to important and/or additional information.
	Tip	Provides helpful information, guidelines, or suggestions for performing tasks more effectively.
	Caution	Warns the user of adverse conditions and/or consequences (e.g., disruptive operations).
	WARNING	Warns the user of severe conditions and/or consequences (e.g., destructive operations).
	DANGER	Dangers provide information about how to avoid physical injury to yourself and others.
	ELECTRIC SHOCK HAZARD!	Warns the user of electric shock hazard. Failure to take appropriate precautions (e.g., do not touch) could result in serious injury.
	ESD Sensitive	Warns the user that the hardware is sensitive to electrostatic discharge (ESD). Failure to take appropriate precautions (e.g., grounded wriststrap) could result in damage to the hardware.

Convention for Storage Capacity Values

Physical storage capacity values (e.g., disk drive capacity) are calculated based on the following values:

1 KB = 1,000 bytes
1 MB = $1,000^2$ bytes
1 GB = $1,000^3$ bytes
1 TB = $1,000^4$ bytes
1 PB = $1,000^5$ bytes

Logical storage capacity values (e.g., logical device capacity) are calculated based on the following values:

1 KB = 1,024 bytes
1 MB = $1,024^2$ bytes
1 GB = $1,024^3$ bytes
1 TB = $1,024^4$ bytes
1 PB = $1,024^5$ bytes
1 block = 512 bytes

Getting Help

If you need to call the Hitachi Data Systems Support Center, make sure to provide as much information about the problem as possible, including the circumstances surrounding the error or failure and the exact content of any messages displayed on the z/OS and/or open-system hosts. Please collect the following information before you call for technical support:

- Error codes: FCU error code, FAL error code, SYS error code. Use the FCU GUI to check recent error information (Help-Error command for UNIX, View-Error information command for NT).
- FCU parameters: direction (mto or otm), input and output files, and FCU options (code conversion, padding, delimiter, empty file, RDW, VSE record).
- FX volume definition file: contents
- FCU parameter definition file (if used): contents
- Command line log (if possible).
- FAL error logs. The FAL logs for UNIX are /tmp/fal_error and /tmp/fal_error.bak, and /tmp/fal_dump and /tmp/fal_dump.bak. The FAL logs for Windows 2000/2003/Windows NT systems are c:\fal_error and c:\fal_error.bak, and c:\fal_dump and c:\fal_dump.bak.
- Windows 2000/2003/Windows NT systems only: FCU log file (e.g., fcudata.prm.log), and Dr. Watson's log file (e.g., c:\WINNT\DRWTSN32.LOG).
- Syslog: error information and other applicable contents

The Hitachi Data Systems customer support staff is available 24 hours/day, seven days a week. If you need technical support, please call:

- United States: (800) 446-0744
- Outside the United States: (858) 547-4526

Comments

Please send us your comments on this document. Make sure to include the document title, number, and revision. Please refer to specific section(s) and paragraph(s) whenever possible.

- **E-mail:** doc.comments@hds.com
- **Fax:** 858-695-1186
- **Mail:**
  Technical Writing, M/S 35-10
  Hitachi Data Systems
  10277 Scripps Ranch Blvd.
  San Diego, CA 92131

Thank you! (All comments become the property of Hitachi Data Systems Corporation.)

1

Overview of Hitachi Cross-OS File Exchange (FX)

This chapter provides a brief overview FX functions, features, and benefits.

The Hitachi Cross-OS File Exchange (FX) feature enables data stored on the TagmaStore® USP subsystems to be converted and transferred between z/OS® and open-system platforms and between different open-system platforms. The FX mainframe-to-open (FXmto) capability enables you to transfer data from z/OS (mainframe) datasets to open-system files. The FX open-to-mainframe (FXotm) capability enables you to transfer data from open-system files to z/OS datasets. The FX open-to-open (FXoto) capability enables you to transfer data between open-system platforms without being attached to a z/OS host. FX utilizes special FX volumes which are dedicated to data exchange operations and are accessed as raw devices to provide the greatest platform flexibility for multiplatform data exchange.

The Hitachi TagmaStore USP and NSC subsystems support concurrent attachment to z/OS hosts and multiple open-system platforms, including UNIX®based platforms (e.g., Solaris, HP-UX, IBM® AIX®) and PC server platforms (e.g., Windows NT, Novell® NetWare®). The TagmaStore USP and NSC subsystems provide high-speed response, continuous data availability, scalable connectivity, and expandable capacity for both z/OS and open-system storage. The TagmaStore USP and NSC subsystem can operate with multihost applications and host clusters, and is designed to handle very large databases as well as data warehousing and data mining applications that store and retrieve terabytes of data.

The Hitachi TagmaStore USP and NSC subsystems provide the following host connectivity options:

- Multiplatform: Both fibre-channel (FC) ports and ExSA™ (ESCON) ports.
- All open: All fibre-channel (FC) ports (no ExSA). This type of subsystem is called an all-open subsystem. FX can be used on the all-open TagmaStore subsystems for FXoto file transfers.

- All mainframe. The TagmaStore subsystems can be configured with all ExSA (no FC ports). This type of subsystem is called an all-mainframe subsystem. FX is not used on the all-mainframe TagmaStore subsystem.

Note: The Hitachi Multiplatform Backup (MBU) feature of the TagmaStore USP and NSC subsystem enables you to implement z/OS-based backup/restore operations (device-level only) for the open-system data stored on the multiplatform TagmaStore USP and NSC. For further information on HMBR, please refer to the *Hitachi Multiplatform Backup (MBU)* (MK-94RD209) or contact your Hitachi Data Systems account team.

Note: For FAL/FCU versions earlier than 01-*XX*-50, Multiple Volume Dataset is not supported. When a dataset is spanned over multiple volumes, only the portion of the dataset within one volume will be processed. For FAL/FCU versions 01-*XX*-50 or later, Multiple Volume Dataset is supported only for FXmto.

Note: FAL/FCU supports two kinds of mainframes, MVS™ and VSE®. For Multiple Volume Dataset, FAL/FCU is only supported on MVS.

Note: When the mainframe OS is VSE, the supported version is 2.3 or earlier.

FX provides the following benefits for the user:

- FX provides a centralized data management and disaster recovery environment for both z/OS and open-system data.
- FX provides high-speed data transfer over ExSA™ (ESCON ®) and fibre channels, freeing up valuable network resources and communication links for application use.
- FX's high-speed data exchange enables you to implement file-level backup of open-system data to z/OS storage (e.g., using Harbor).

2

About Cross-OS File Exchange Operations

This chapter describes how FX operates in typical system configurations and describes the operations a user may perform using FX.

- ☐ Components
- ☐ FCU File Transfer Options
- ☐ FXmto Operations
- ☐ FXotm Operations
- ☐ FXoto Operations
- ☐ Host Access and I/O Contention
- ☐ AIX Shared Open Function
- ☐ AIX Reserve Function
- ☐ Bidirectional Data Transfer

Components

Figure 2-1 illustrates the typical system configurations required for FXmto and FXotm operations. FXmto/otm operations are performed using the File Conversion Utility (FCU) and File Access Library (FAL), which are installed on the open-system host(s).

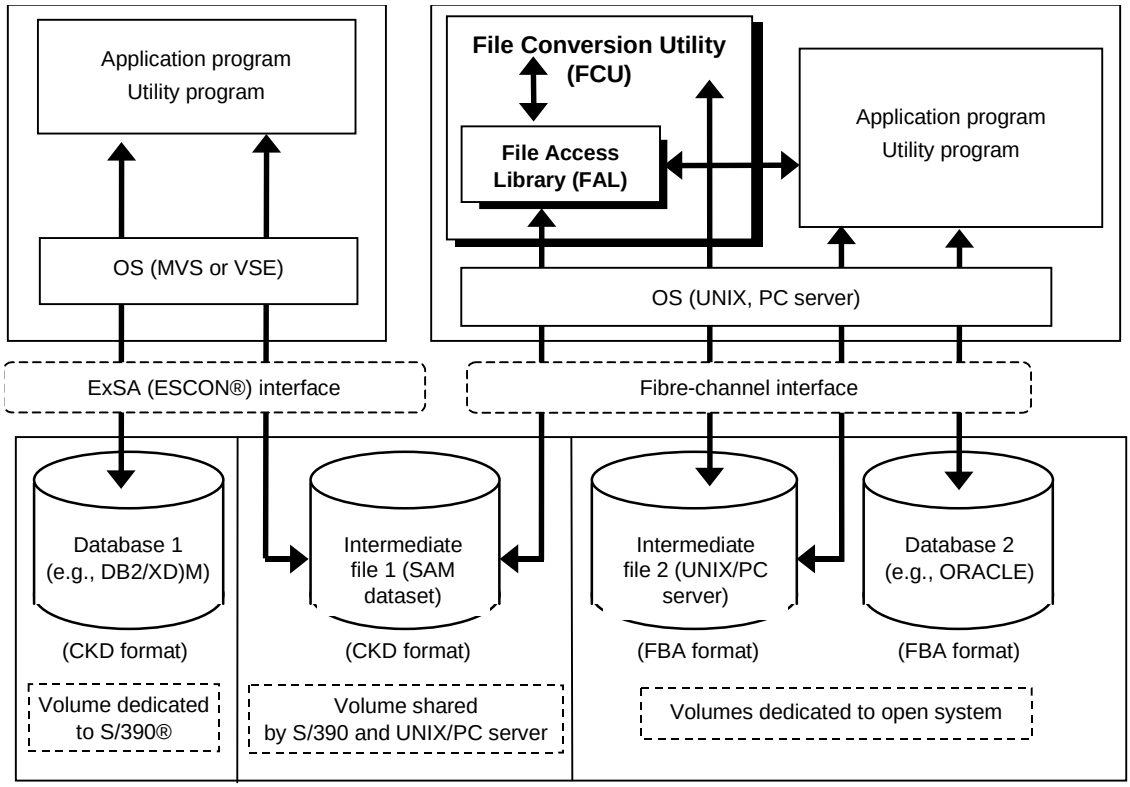


Figure 2-1 FXmto and FXotm System Configuration

Figure 2-2 illustrates the typical system configurations required for FXoto operations, which are performed using the Formatter (FMT) and Allocater (ALC) utilities in addition to FCU and FAL.

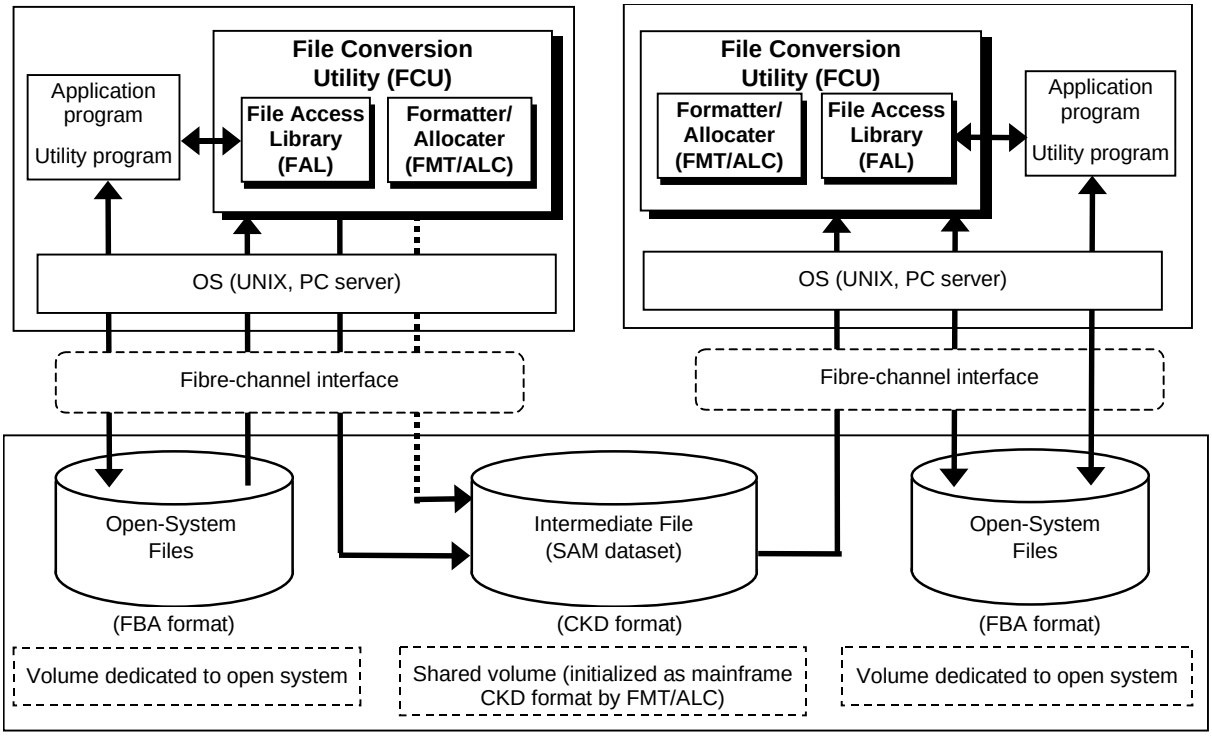


Figure 2-2 FXoto System Configuration

The FCU program provides the commands and graphical user interface (GUI) for FX file transfer operations as well as several important options for data exchange, including EBCDIC-ASCII code conversion and data record padding and delimiters. The FAL is a library of C functions (Visual C++® for Windows NT®), which provides an application-programming interface for data exchange. The FAL functions can be called by user application programs to read and write data in z/OS datasets on the TagmaStore USP and NSC. There are two types of FAL, the 32bitFAL and the 64bitFAL. The latter is provided by the TagmaStore USP and NSC. The FMT and ALC utilities enable the all-open TagmaStore USP and NSC user to format OPEN-*x* logical units (LUs) and create intermediate datasets for FXoto operations, without having to be attached to any z/OS hosts. OPEN-*x* is defined as a standard LU type. The TagmaStore USP and NSC subsystem currently supports OPEN-3, OPEN-8, OPEN-9, OPEN-E, OPEN-L and OPEN-V devices. Please contact your Hitachi Data Systems account team for the latest information on supported LU types.

FX Volume Types

The FXmto and FXotm volumes are z/OS devices that can only be accessed by open-system hosts using the FAL/FCU software. The FXoto volumes are open-system devices that cannot be accessed by z/OS hosts. FX operations are performed using the following types of FX volumes on the TagmaStore USP and NSC subsystems:

- **FXmto**, **FXotm**. The FX -A volumes can be used for FXmto and FXotm operations. z/OS hosts have normal read/write access to -A volumes. Open-system hosts have read/write access to -A volumes but must use FAL/FCU to access these volumes as raw devices (no mount operation). Figure 2-3 shows the structure of the FX -A volumes.

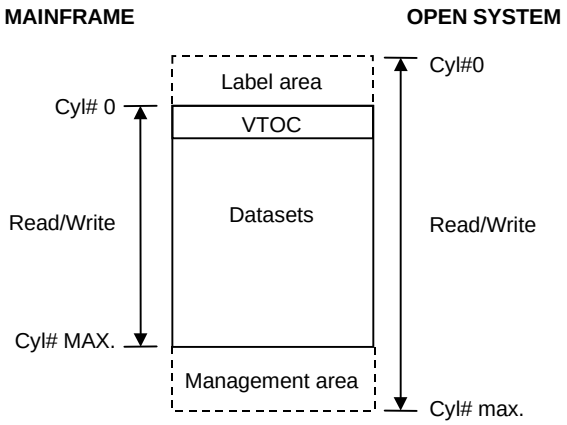


Figure 2-3 3390-3A/9A/LA Volume Structure (FXmto, FXotm, FXoto)

Note: The -A volumes are not write-protected. Do not execute any open-system write operations to -A volumes (except disk partitioning and labeling). Do not create a file system on an -A volume; this will overwrite the data exchange files on the volume.

- **FXmto**. The FX -B volumes can only be used for FXmto operations. z/OS hosts have normal read/write access to -B volumes. Open-system hosts have read-only access to -B volumes and must use FAL/FCU to read these volumes as raw devices (no mount operation). The -B volumes are write-protected from open-system access. The TagmaStore subsystem will reject all open-system write operations to -B volumes (except disk partitioning and labeling) to protect the z/OS data on these volumes. Figure 2-4 shows the structure of the FX -B volumes.

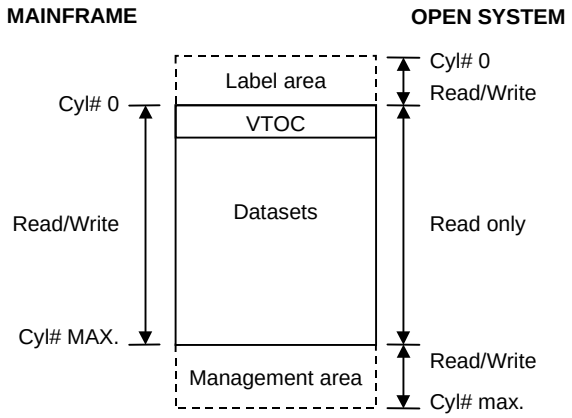


Figure 2-4 3390-3B/9B/LB Volume Structure (FXmto)

Note: The open-system host accesses only the volume table of contents (VTOC) area on -B volumes. Catalog or security control functions cannot be used to provide access control for these volumes.

- **FXotm**. The FX -C volumes can only be used for FXotm operations. Open-system hosts have read/write access to the -C volumes but must use FAL/FCU to access these volumes as raw devices (no mount operation). z/OS hosts have read-only access to the -C volumes. The TagmaStore subsystem will reject all z/OS write operations to -C volumes (except VTOC) to protect the open-system data on these volumes. Figure 2-5 shows the structure of the FX -C volumes.

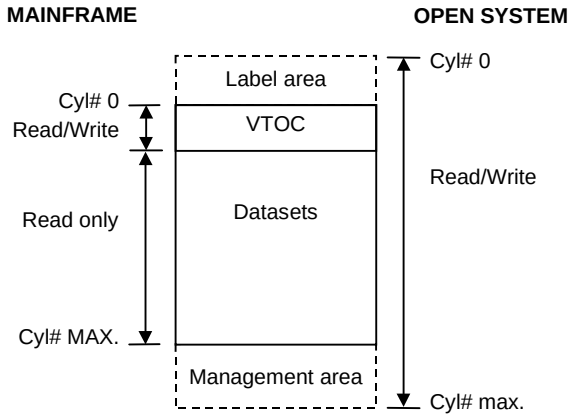


Figure 2-5 3390-3C/9C/LC, 3380-KC/3C Volume Structure (FXotm)

- **FXoto**. OPEN-*x* volumes that are formatted with the FX Formatter (FMT) utility can only be used for FXoto operations. Open-system hosts have read/write access to the OPEN-*x* FMT volumes but must use FAL/FCU to access these volumes as raw devices (no mount operation). z/OS hosts do not have any access to the OPEN-*x* FMT volumes. Figure 2-6 shows the structure of the FX OPEN-*x* FMT volumes.

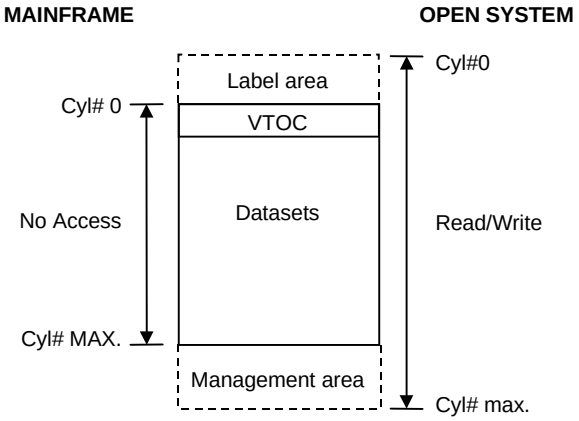


Figure 2-6 OPEN-*x* FMT Volume Structure (FXoto)

Note: When the mainframe OS is VSE, DFSORT cannot be used after an FX volume is copied to another volume. Use the DITTO function for this purpose.

Note for AIX: When installing FX devices in an AIX environment, the following notes apply:

- When the UserID is not the root, a patch is required.
- AIX V5.1 can be used with 64bitFAL.

Please contact your IBM technical representative for assistance.

Note for Microsoft Cluster Server: When installing FX devices in a Microsoft Cluster Server (MSCS) environment, you must write signatures on the FX volumes before configuring MSCS.

- The MSCS server cannot connect volumes that do not have signatures.
- The volume on which a signature is written cannot be accessed from another server.
- The volume on which a signature is written cannot be shared.
- Only the mainframe and the server that wrote the signature can access the volume that has the signature.
- Signatures cannot be written to FX volumes for which the emulation type is 3390-3*X*, 3390-9*X* 3390-L*X* , or 3380-K*X*, 3380-3*X* (*X* = A, B, C), when the OS server is Windows 2000/2003/Windows NT.
- When configuring MSCSand the server OS is Windows 2000/2003/Windows NT, FXotm and FXmto cannot be started.

Notes for Windows 2000/2003/Windows NT:

- Service Pack 1 must be installed when MSCS is configured
- A signature is not necessary for the MSCS configuration. A Write Error will occur if a signature is attempted.

Warning: Do not write a signature on FX Volumes having emulation types 3390-3X, or 3390-9X, 3390-LX, or 3380-KX, 3380-3X (X=A,B,C) in a Windows 2000/2003/Windows NT environment. If a signature-writing attempt is made by the Disk Administrator with Windows 2000/2003/Windows NT, a Write Error will appear in order to stop the signature from being written. When the Windows 2000/2003/Windows NT Disk Administrator starts again, a request will be made again to write the signature. Do not write the signature.

FCU File Transfer Options

For each FX operation, FCU requires that the transfer direction (mto or otm) and the source and target files be identified. (An FXoto operation consists of one FXotm operation followed by one FXmto operation.) In addition to these required parameters, FCU provides the following options for FX file transfer operations:

- Code conversion (CC)
- Padding (PAD)
- Delimiters (DEL)
- Record description word (RDW)
- VSE record (VSE)
- Empty file (Emp)

Note: For FAL/FCU versions earlier than 01-*XX*-50, Multiple Volume Dataset is not supported. When a dataset is spanned over multiple volumes, only the portion of the dataset within one volume will be processed. For FAL/FCU versions 01-*XX*-50 or later, Multiple Volume Dataset is supported only for FXmto.

Note: FAL/FCU supports two kinds of mainframes, MVS and VSE. For Multiple Volume Dataset, FAL/FCU is only supported on MVS.

Code Conversion (CC) Option

The code conversion option can be used for FXmto and FXotm operations. The code conversion option enables you to specify either the default EBCDIC-ASCII code conversion table included with FCU (see Table 2-1), or your own code conversion table (see Table 2-2). When the default table is specified, FCU performs EBCDIC-to-ASCII code conversion for FXmto operations and ASCII-to-EBCDIC code conversion for FXotm operations as specified in Table 2-1 (see also Appendix C). The user-defined code conversion table must be a binary data file created by placing the target code values in the offset positions that correspond to the source code values.

Always use code conversion when transferring text files between mainframe and open systems. Do not use code conversion when transferring binary data files. Code conversion is available (**EcA** option) but not recommended for FXoto file transfers.

Note: FCU versions 01-01-40 and later support the user-defined code conversion table.

Note: The default EBCDIC-ASCII code conversion table is the ACM standard table (not CACM). Appendix C provides the code conversion information for the default table which is shown in Table 2-1. If the default code conversion table does not yield the desired results, create your own code conversion table. Please refer to the IBM code tables for detailed information on EBCDIC-ASCII code conversion.

Table 2-1 Default EBCDIC-ASCII Code Conversion Table for FCU

H L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL (00)	DLE (10)	DS (80)	(90)	SP (20)	& (26)	- (2D)	(BA)	(C3)	(CA)	(D1)	(D8)	{ (7B)	} (7D)	\ (5C)	0 (30)
1	SOH (01)	DC1 (11)	SOS (81)	(91)	(A0)	(A9)	/ (2F)	(BB)	a (61)	j (6A)	(E5)	(D9)	A (41)	J (4A)	(9F)	1 (31)
2	STX (02)	DC2 (12)	FS (82)	SYN (16)	(A1)	(AA)	(B2)	(BC)	b (62)	k (6B)	s (73)	(DA)	B (42)	K (4B)	S (53)	2 (32)
3	ETX (03)	DC3 (13)	(83)	(93)	(A2)	(AB)	(B3)	(BD)	c (63)	l (6C)	t (74)	(DB)	C (43)	L (4C)	T (54)	3 (33)
4	PF (9C)	TM (9D)	BYP (84)	PN (94)	(A3)	(AC)	(B4)	(BE)	d (64)	m (6D)	u (75)	(DC)	D (44)	M (4D)	U (55)	4 (34)
5	HT (09)	(85)	LF (0A)	RS (95)	(A4)	(AD)	(B5)	(BF)	e (65)	n (6E)	v (76)	(DD)	E (45)	N (4E)	V (56)	5 (35)
6	LC (86)	BS (08)	ETB (17)	UC (96)	(A5)	(AE)	(B6)	(C0)	f (66)	o (6F)	w (77)	(DE)	F (46)	O (4F)	W (57)	6 (36)
7	DEL (7F)	IL (87)	ESC (1B)	EOT (04)	(A6)	(AF)	(B7)	(C1)	g (67)	p (70)	x (78)	(DF)	G (47)	P (50)	X (58)	7 (37)
8	GE (97)	CAN (18)	(88)	(98)	(A7)	(B0)	(B8)	(C2)	h (68)	q (71)	y (79)	(E0)	H (48)	Q (51)	Y (59)	8 (38)
9	RLF (8D)	EM (19)	(89)	(99)	(A8)	(B1)	(B9)	‘ (60)	i (69)	r (72)	z (7A)	(E1)	I (49)	R (52)	Z (5A)	9 (39)
A	SMM (8E)	CC (92)	SW (8A)	(9A)	(D5)	! (21)	(CB)	: (3A)	(C4)	^ (5E)	(D2)	(E2)	(E8)	(EE)	(F4)	(FA)
B	VT (0B)	CUI (8F)	CUI (8B)	CU3 (9B)	. (2E)	$ (24)	, (2C)	# (23)	(C5)	(CC)	(D3)	(E3)	(E9)	(EF)	(F5)	(FB)
C	FF (0C)	IFS (1C)	(8C)	DC4 (14)	< (3C)	* (2A)	% (25)	@ (40)	(C6)	(CD)	(D4)	(E4)	(EA)	(F0)	(F6)	(FC)
D	CR (0D)	IGS (1D)	ENQ (05)	NAK (15)	((28)	) (29)	_ (5F)	’ (27)	(C7)	(CE)	[(5B)	] (5D)	(EB)	(F1)	(F7)	(FD)
E	SO (0E)	IRS (1E)	ACK (06)	(9E)	+ (2B)	; (3B)	> (3E)	= (3D)	(C8)	(CF)	(D6)	(E6)	(EC)	(F2)	(F8)	(FE)
F	SI (0F)	IUS (1F)	BEL (07)	SUB (1A)	| (7C)	~ (7E)	? (3F)	“ (22)	(C9)	(D0)	(D7)	(E7)	(ED)	(F3)	(F9)	(FF)

Note: Appendix C shows the EBCDIC-ASCII code conversion information for this binary table.

Legend for Table 3-1

	Bit Positions	
	Hi	Lo
ASCII	8765	4321

EBCDIC (IBM)	0123	4567

Table 2-2 User-Defined Code Conversion Table

Item	Requirement(s)
Size	256 bytes
Format	Binary data
Code length	One byte (two-byte codes cannot be converted)
File name	The following sequences of characters cannot be used in the file name: EA EcA EkJ No If the file name for the code conversion table contains any of these sequences, FCU will ignore the file and use the default table instead.

PIPE Function

This function transfers data entries from the mainframe to the application program or the utility program for UNIX systems using a “named pipe”. When this function is used, a mainframe dataset can be transferred to an open system. This is a much faster way to transfer data than the Code Conversion method.

A "named pipe" is a special file that is used to transfer data between unrelated processes. One (or more) processes writes to it, while another process reads from it. Named pipes are visible in the file system and may be viewed with ‘**ls**’ like any other file. (Named pipes are also called "fifo"s; this term stands for `First In, First Out'.) Named pipes may be used to pass data between unrelated processes, while normal (unnamed) pipes can only connect parent/child processes (with some exceptions). Named pipes are strictly unidirectional, even on systems where anonymous pipes are bidirectional (full-duplex).

Using the PIPE function in UNIX Systems

A "named pipe" is a special file that is used to transfer data between unrelated processes. One (or more) processes write to it, while another process reads from it. Named pipes are visible in the file system and may be viewed with `ls' like any other file. (Named pipes are also called "fifo"s; this term stands for `First In, First Out'.) Named pipes may be used to pass data between unrelated processes, while normal (unnamed) pipes can only connect parent/child processes (unless you try *very* hard). Named pipes are strictly unidirectional, even on systems where anonymous pipes are bidirectional (full-duplex).

FAL/FCU Pipe Function Details

FCU can carry out data transmission to a pipe file. A user application opens and reads this pipe file, and a direct data transmission is attained between the application and FCU. There is no need for it to be output as a file on an HDD.

A named-pipe name is specified in a parameter definition file as the output file name. **"PIPE=Yes"** needs to be specified as an option. Under these conditions, FCU will open a pipe file with the specified output file name, and will transmit data to it.

If the specified file exists as a standard UNIX file, FCU re-creates a pipe file using the same name (the UNIX file is deleted.) Since FCU only inputs in data to a pipe, the FCU function needs to obtain the data via a user application. If data remains in the pipe, FCU will stop and processing does not progress to the next step. The FCU function has an inbuilt timer. If the application does not continue receiving data, FCU will send an error message after a certain set time, and it will progress to the next logical process.

Note: This function is only supported for UNIX systems. It is not supported for Windows 2000/2003/Windows NT. It is supported for mainframe to open systems data transfer only. This function requires an application program or a utility program to receive data entries using a named pipe.

Pipe Function Time-Out Value

- FCU waits for a "Read Data Entries" status message. A time-out error will be reported if the TIME OUT VALUE is not set appropriately. The TIME OUT VALUE should be set in the WAIT_TIME_VALUE environment variable. The limits are 0~1440 seconds (0 = unlimited). The default value is 10 when the timeout value is undefined.
- The following examples illustrate the use of the WAIT_TIME_VALUE environment variable.

Note: After setting the variable, log out and log in again to establish the variable's value.

- ***Example 1:*** For C shell:
  - Add "setenv WAIT_TIME_VALUE 300" to the file".cshrc" in the home directory.
  - If ".cshrc" does not exist, create it and add the "setenv" line.
- ***Example 2:*** For non-C shell:
  - Add "WAIT_TIME_VALUE=300"
  - Add "export WAIT_TIME_VALUE"

  These two commands must be added to the file ".dtprofile" in the home directory. If ".dtprofile" does not exist, create it and add the lines.

Figure 2-7 illustrates the Pipe function process.

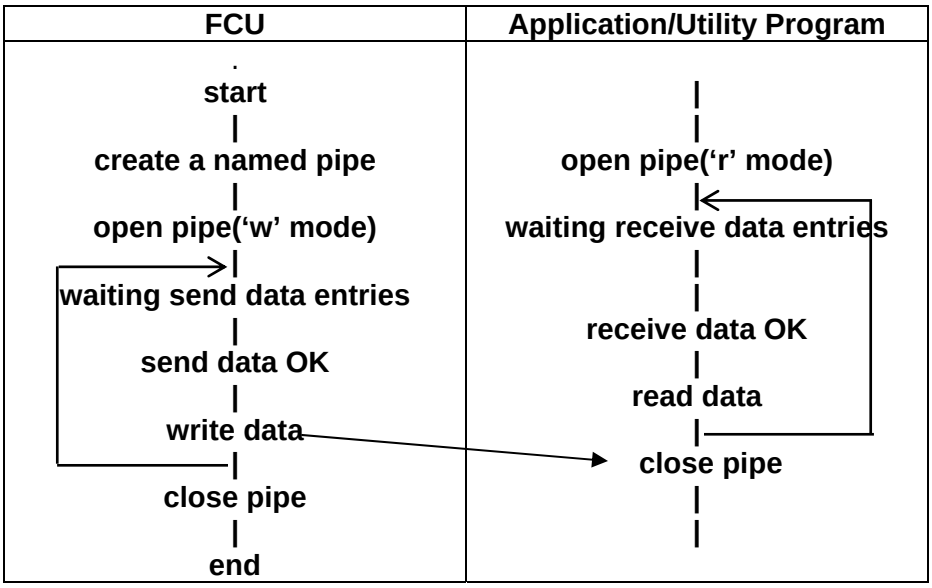


Figure 2-7 Pipe Function Process Outline

Padding (PAD) Option

The padding option can be used for FXmto with variable-length source datasets and for FXotm with fixed-length target datasets. When the padding option is specified for FXmto, FCU adds padding to each source data record, so that the length of the each record equals the maximum record length. When padding is specified for FXotm, FCU adds padding to each source data entity, so that the length of the each target record equals the record length defined for the target dataset. FCU transfers the data entities with padding to the target file/dataset. FCU cannot extract padding from files or datasets. Sections FXmto Operations and FXmto with Multiple Volume Datasets describe FXmto and FXotm operations with padding.

The type of padding added by FCU depends on whether code conversion was also requested:

- **Padding with code conversion (text files)**. When padding and code conversion are both specified, FCU adds **spaces** to the short data entities as needed.
- **Padding without code conversion (binary data files)**. When padding is specified but code conversion is not, FCU adds **0x00** to the short data entities as needed.

Note: If you use FXmto with padding, the data cannot be transferred back to the original z/OS dataset (the FXotm target dataset will not be compatible with the original dataset). If you use FXotm with padding, the delimiter option is required.

Delimiter (DEL) Option

The delimiter option can be used for both FXmto and FXotm operations and enables variable-length records to be transferred between platforms without losing compatibility with the original dataset. When the delimiter option is specified for FXmto, FCU adds the specified delimiter to the end of each data entity in the source file, and then extracts and transfers the data entity with delimiter to the open-system target file. When the delimiter option is specified for FXotm, FCU extracts each data entity preceding the specified delimiter and transfers the data entities without delimiters to the target dataset. Sections FXmto Operations and FXmto with Multiple Volume Datasets describe FX operations with delimiters.

The type and length of the delimiter added (or recognized and extracted) by FCU depends on the open-system platform:

- For UNIX-based platforms, you must specify either a carriage return (CR) or a line feed (LF). The length of this delimiter is one byte.
- For Windows 2000/2003/Windows NT, you must specify a CR + LF. The length of this delimiter is two bytes.

Note: Do not use the delimiter option for FXotm if the source file contains the same character(s) as the delimiter but used for a purpose other than delimiting data entities. If you do, FCU will interpret the specified delimiter character(s) as delimiters, which can create a target dataset with corrupt records or generate an error condition.

Note: If you use FXmto with delimiter (no padding) for variable-length records, you will be able to transfer the data back to the original z/OS dataset later using FXotm.

Empty File (Emp) Option

The empty file (Emp) option can be used for both FXmto and FXotm operations. When the empty file option is specified, FCU processes an empty source file instead of returning an error. An empty z/OS dataset is a dataset which has no records or only EOF records. An empty open-system file is a file which has a file size of 0 bytes. When an empty z/OS dataset is processed, the open-system target file size = 0. When an empty open-system file is processed, the target dataset will contain only EOF records.

Note: FCU version 01-01-38 or later for UNIX and FCU version 01-01-36 or later for Windows 2000/2003/Windows NT are required for the empty file option.

Record Description Word (RDW) Option

The record description word option can only be used for FXmto operations on variable-length source datasets. Figure 2-8 shows an FXmto operation with the RDW option specified. When the RDW option is specified, FCU adds the record description word in binary code to the head of each record in the source dataset, and then transfers the data entity with record length bytes to the open-system target file. The CC, PAD, and DEL parameters must be **No**; if not, FCU returns an error. If the RDW option is specified for a fixed-length source dataset, FCU ignores the RDW option.

Note: If you use FXmto with RDW, the data cannot be transferred back to the original z/OS dataset (the FXotm target dataset will not be compatible with the original dataset).

Note: FCU version 01-01-38 or later for UNIX and FCU version 01-01-36 or later for Windows 2000/2003/Windows NT are required for the RDW option.

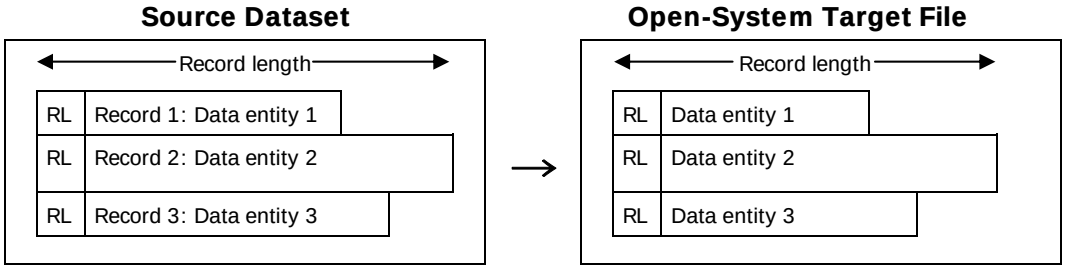


Figure 2-8 FXmto with the RDW Option

VSE Record (VSE) Option

The VSE record option must be used for FXmto and FXotm operations involving VSE datasets. The VTOC of a VSE dataset does not specify the record format (RF), record length (RL), or block length (BL) of the dataset. The VSE record option enables the user to specify these values so that FCU can process source/target VSE datasets. If you do not specify the VSE record option for a VSE dataset, FCU will return an error. If you specify the VSE record option and the RF, RL, and BL are also specified in the VTOC, FCU will process the dataset if the RF, RL, and BL values are the same, or return an error if the RF, RL, and BL values are not the same. The VSE record option does not apply to ALC-generated intermediate datasets.

Note: FCU version 01-01-40 or later is required for the VSE record option.

FXmto Operations

An FXmto operation transfers the data from a z/OS dataset on an FX volume to an open-system file on an open-system LU. The object data entities are those contained in all records between the beginning of the file and the end of the file. The end of a dataset is the EOF record or the end of the final extent. The end of an open-system file is the EOF. The FXmto source file must be located on an FX -B or -A volume on the subsystem. If the specified FXmto target file does not exist, FCU automatically creates the target file during the FXmto operation. If the specified FXmto target file already exists, FCU requests confirmation to overwrite the target file (unless the **-nc** option is specified).

The FCU software performs the FXmto data transfer operations. FCU version 01-01-40 or later is required for VSE source datasets. FCU supports both fixed-length and variable-length record formats and provides the following options for FXmto data transfer (see section entitled “System Requirements”): code conversion, padding, delimiter, empty file, record description word, and VSE record. The types of FXmto operations are:

- FXmto with fixed-length record format
- FXmto with variable-length record format)

Table 2-3 specifies the record format requirements for each type of FXmto operation. A fixed-length source dataset can only be transferred to a fixed-length target file, with or without delimiters. Padding cannot be added to a fixed-length source file. A variable-length source dataset can be transferred to a variable-length or fixed-length target file, depending on the padding option, and delimiters can also be added if desired.

Table 2-3 FXmto Record Format Requirements

FCU Direction	Padding	Delimiters	Record Format Requirements		See Figure:
			Source Dataset	Target File	
FXmto	N/A	No	Fixed-length →	Fixed-length	3.2
FXmto	N/A	Yes	Fixed-length →	Fixed-length	3.3
FXmto	No	No	Variable-length →	Variable-length	3.4
FXmto	Yes	No	Variable-length →	Fixed-length	3.5
FXmto	No	Yes	Variable-length →	Variable-length	3.6
FXmto	Yes	Yes	Variable-length →	Fixed-length	3.7

FXmto with Fixed-Length Record Format

Each fixed-length record in a z/OS dataset includes only the fixed-length data entity. The record length defined for a fixed-length dataset equals the actual length of each data entity. The padding option cannot be used for FXmto with fixed-length records.

No padding, no delimiters. Figure 2-9 shows an FXmto operation for a fixed-length source dataset. Padding cannot be added to fixed-length records. FCU extracts and transfers the data entities to the open-system target file. The length of each data entity in the target file equals the record length defined for the source dataset.

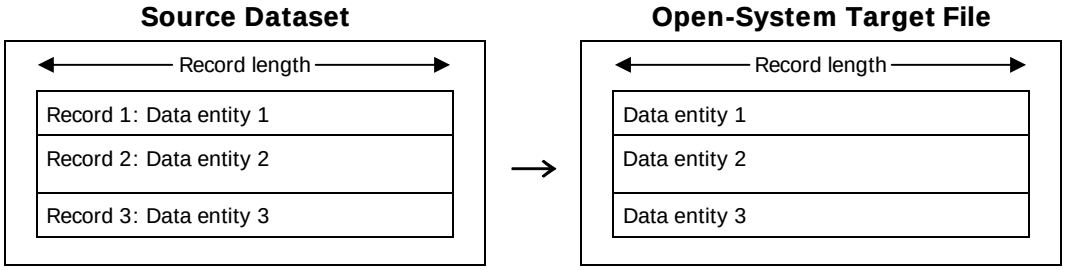


Figure 2-9 FXmto with Fixed-Length Records: No Padding, No Delimiters

With delimiters. Figure 2-10 shows an FXmto operation with delimiters (D) for a fixed-length source dataset. FCU extracts and transfers the data entities to the open-system target file and adds the requested delimiter to the end of each data entity. The resulting length of each data entity in a UNIX target file equals the original record length plus one byte for the delimiter. The resulting length of each data entity in a Windows 2000/2003/Windows NT target file equals the original record length plus two bytes for the delimiter.

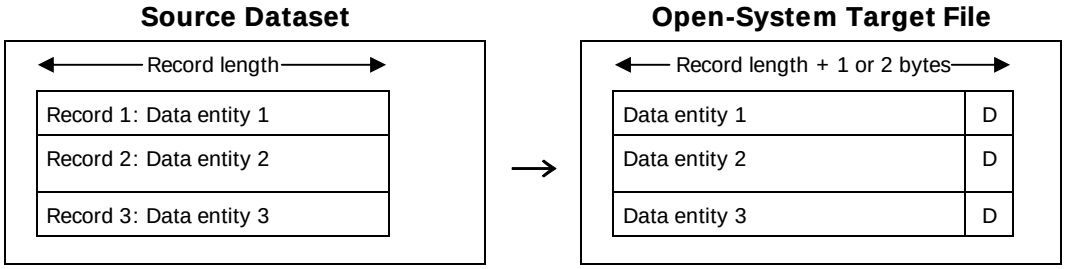


Figure 2-10 FXmto with Fixed-Length Records: Delimiters

FXmto with Variable-Length Record Format

Each variable-length record in a z/OS dataset includes a four-byte RL field and the variable-length data entity. The record length defined for a variable-length dataset equals the maximum allowable record length.

Note: If you want to be able to transfer the data back to the original z/OS dataset later, you must use FXmto without padding and with delimiters.

No padding, no delimiters. Figure 2-11 shows an FXmto operation without padding or delimiters for a variable-length source dataset. FCU extracts and transfers only the data entities to the target file. The RL fields are not transferred. The resulting length of each data entity in the target file is equal to or less than the maximum record length minus four bytes (for the RL field).

Note: If you plan to transfer the data back to the original dataset later using FXotm, use FXmto with delimiters.

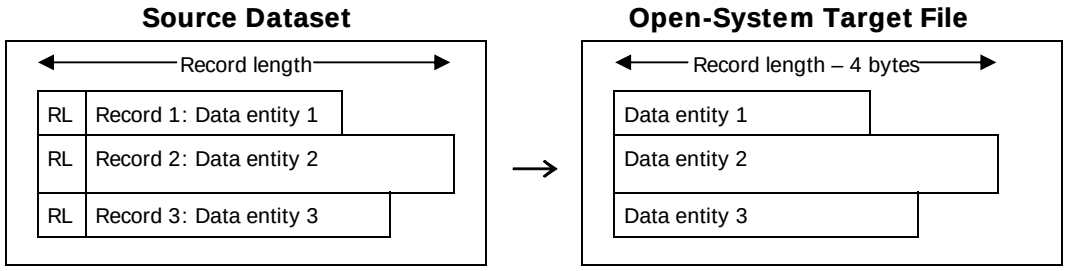


Figure 2-11 FXmto with Variable-Length Records: No Padding, No Delimiters

With padding. Figure 2-12 shows an FXmto operation with padding. FXmto with padding requires a variable-length source file and produces a fixed-length target file. FCU adds padding to the source records as needed so that the length of each record equals the maximum record length. FCU then extracts and transfers the data entities with padding to the open-system target file. The RL fields are not transferred. The resulting length of each data entity in the target file equals the maximum record length minus four bytes (for the RL field).

Note: If you use FXmto with padding, you will not be able to transfer the data back to the original dataset later using FXotm.

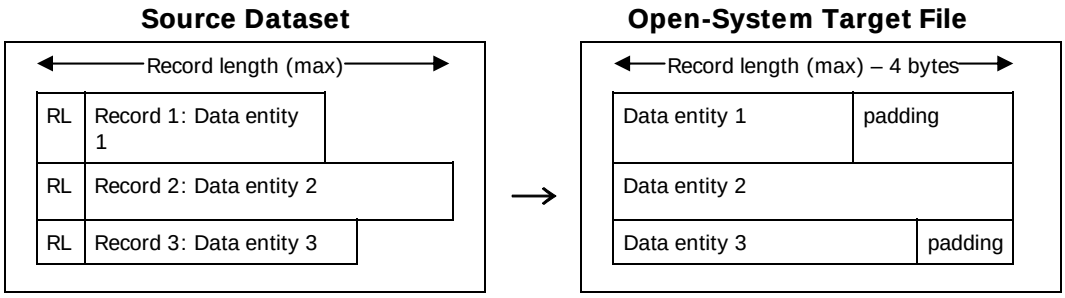


Figure 2-12 FXmto with Variable-Length Records: Padding

With delimiters. Figure 2-13 shows an FXmto operation with delimiters (D) for a variable-length source dataset. FCU extracts and transfers the data entities to the open-system target file and adds the requested delimiter to the end of each data entity. The RL fields are not transferred. The resulting length of each data entity in a UNIX target file equals the original data entity length plus one byte for the delimiter. The resulting length of each data entity in a Windows 2000/2003/ Windows NT target file equals the original data entity length plus two bytes for the delimiter.

Note: If use FXmto with delimiters and without padding, you will be able to transfer the variable-length records back to the original dataset later using FXotm.

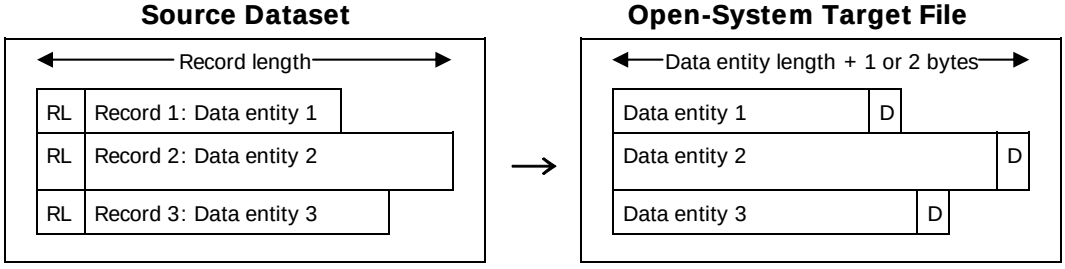


Figure 2-13 FXmto with Variable-Length Records: Delimiters

With padding and delimiters. Figure 2-14 shows an FXmto operation with padding and delimiters (D). FXmto with padding and delimiters requires a variable-length source file and produces a fixed-length target file. FCU adds the appropriate delimiter to each data entity, adds the appropriate amount of 'padding' so that each record equals the maximum record length, and then extracts and transfers the data entities with padding and delimiters to the open-system target file. The RL fields are not transferred.

Note: If you use FXmto with padding and delimiters, you will not to be able to transfer the records back to the original dataset later (the padding cannot be removed).

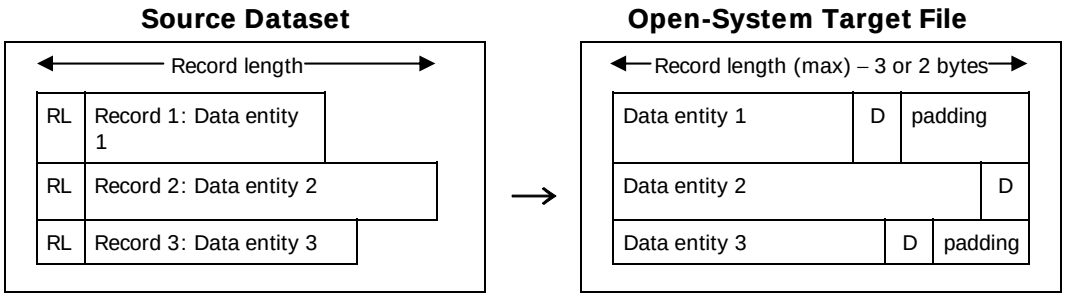


Figure 2-14 FXmto with Variable-Length Records: Padding and Delimiters

The resulting length of each data entity in a UNIX target file equals the maximum record length minus three bytes (minus four for the RL, plus one for the delimiter). The resulting length of each data entity in a Windows 2000/2003/Windows NT target file equals the maximum record length minus two bytes (minus four for the RL, plus two for the delimiter).

FXmto with Multiple Volume Datasets

Multiple Volume Dataset is supported only for FXmto, when the version is 01-XX-50/ YY(XX=01 or 02, 03 or 04) or higher. Table 2-4 Illustrates this.

Note: Multiple Volume Dataset is not supported for FXotm. A multiple volume definition file (multidef.dat) is necessary in current directory. FAL will check Dataset Serial number, Data set serial number and Last volume containing data in this data set in Data set indicators on VTOC DSCB1.

Note: Multiple volume Dataset is supported for FileExchangeotm when the version is 01-XX-60/YY (XX=02 or 03 or 04) or higher.

Note: For Windows 2000/2003, you should not write a signature on shared volumes, which are 3390-3X, 3390-9X, 3390-LX, or 3380-KX, 3380-3X (X=A,B,C). If you try to write a signature on the shared volumes, FX can not guarantee that the volumes will be shared with other OS's (AIX, Solaris, HP-UX, Tru64, Linux, etc). When you use a shared volume with Windows 2000/2003 only, FX will perform correctly if you write a signature on the shared volumes. (A "write error" message will appear in the System log, but this will not have a negative influence on FX operation.)

Note: FAL/FCU supports two kinds of mainframes, MVS™ and VSE. For Multiple Volume Dataset, FAL/FCU is only supported on MVS™.

Table 2-4 Multiple Volume Dataset/Serial Numbers

VTOC DSCB1/ Action	Data Set Serial Number			
	1	1	Except 1	Except 1
	Last Volume Containing Data in this Data Set, in Data Set Indicators			
	On	Off	On	Off
OTM	OK(OK)	NG(OK)	NG(OK)	NG(OK)
MTO for single volume(*1)	OK(OK)	NG(OK)	NG(OK)	NG(OK)

FXotm Operations

An FXotm operation transfers the data from an open-system file on an FX volume to a target dataset on an z/OS volume. The FXotm source file must be located on an FX -C or -A volume on the subsystem. FCU does not automatically create the FXotm target dataset. The target dataset must be created and properly formatted prior to beginning the FXotm operation.

The FCU software performs the FXotm data transfer operations. FCU version 01-01-40 or later is required for VSE target datasets. FCU supports fixed-length and variable-length record formats for FXotm operations. FCU provides the following options for FXotm operations (see section FCU File Transfer Options): code conversion, padding (01-01-41 and later), delimiter, empty file, and VSE record. The record description word option cannot be used with FXotm. FCU automatically extracts delimiters from FXotm source files, but cannot add delimiters to FXotm source files. FCU can add padding only to variable-length FXotm source files. FCU cannot extract padding from FXotm source files. The types of FXotm operations are:

- FXotm with fixed-length record format
- FXotm with variable-length record format

The table below specifies the record format requirements for each type of FXotm operation. An open-system source file with fixed-length data entities can only be transferred to a fixed-length target dataset. An open-system source file with variable-length data entities must have delimiters and can be transferred to a variable-length or fixed-length target dataset. If the source file contains padding from a previous FXmto transfer operation, the padding is transferred to the target dataset along with the data. If the source file contains delimiters, the delimiters are not transferred to the target dataset.

Note: Do not update the volume that is transferred directly by the FXotm.

Table 2-5 FXotm Record Format Requirements

FCU Direction	Record Format Requirements		See Figure:
	Source File	Target Dataset	
FXotm	Fixed-length: no padding, no delimiters →	Fixed-length	3.8
FXotm	Fixed-length containing padding →	Fixed-length	3.9
FXotm	Fixed-length containing delimiters →	Fixed-length	3.10
FXotm	Fixed-length containing padding and delimiters →	Fixed-length	3.11
FXotm	Variable-length: with delimiters →	Variable-length	3.12
FXotm	Variable-length: with padding and delimiters →	Fixed-length	3.13

FXotm with Fixed-Length Record Format

No padding, no delimiters. The figure below shows an FXotm operation for a fixed-length source file without padding or delimiters. The target dataset must have fixed-length record format with record length set to the actual length of each data entity. If the data entity length does not exactly match the record length defined for the target dataset, FCU aborts the operation and reports an error.

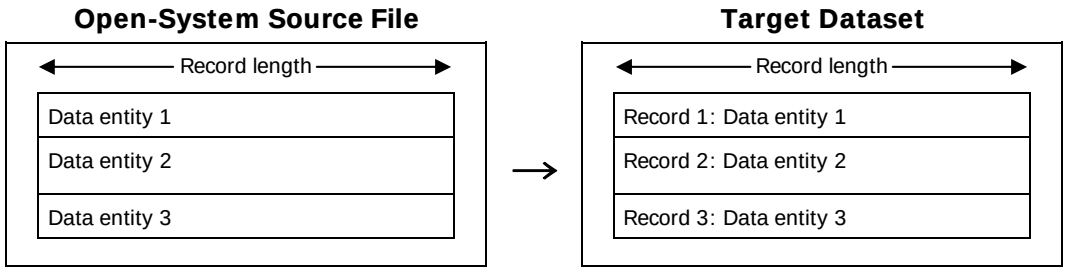


Figure 2-15 FXotm with Fixed-Length Records: No Padding, No Delimiters

With padding. The figure below shows an FXotm operation for a fixed-length source file with padding from a previous FXmto transfer. The original FXmto dataset cannot be used as the FXotm target dataset. FCU transfers the data entities including padding to the target dataset. The length of each data entity in the source file equals the maximum record length minus four bytes (for the RL field). The target dataset must have fixed-length record format with record length set to the maximum record length minus four bytes. If the length of any record (data entity plus padding) in the source file does not exactly match the record length defined for the target dataset, FCU aborts the operation and reports an error.

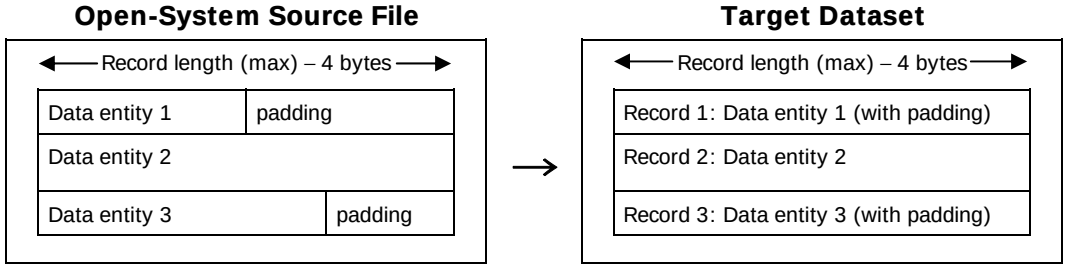


Figure 2-16 FXotm with Fixed-Length Records: Padding

Note: FCU does not extract padding from FXotm source files.

With delimiters. The figure below shows an FXotm operation for a fixed-length source file with delimiters from a previous FXmto transfer. FCU extracts the data entities from the source file by record length and transfers them to the target dataset. The delimiters are not transferred. The target dataset must have fixed-length record format with record length set to the actual length of each data entity (without delimiter). If the length of any source data entity does not exactly match the record length defined for the target dataset, FCU aborts the operation and reports an error. If the delimiter is not found right after the data entity, FCU aborts the operation reports an error.

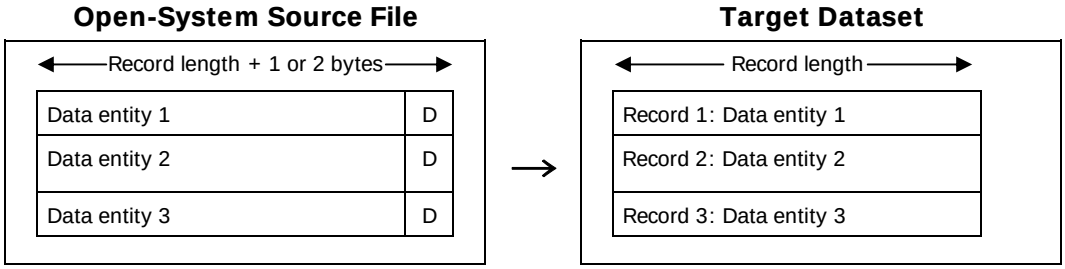


Figure 2-17 FXotm with Fixed-Length Records: Delimiters

Note: FCU does not add delimiters to FXotm source files. If the FXotm source file contains delimiters but you specify **No** for the delimiter option, the delimiters will be regarded as part of the data entities and will be transferred to the target dataset.

With padding and delimiters. The figure below shows an FXotm operation for a fixed-length source file with padding and delimiters from a previous FXmto transfer. FCU removes the delimiters but not the padding and transfers the data entities with padding to the target dataset. The original variable-length dataset cannot be used as the target dataset for this transfer. The target dataset must have fixed-length record format with record length set to the maximum record length minus four bytes. If the length of any source data entity does not match the record length defined for the target dataset, FCU aborts the operation and reports an error.

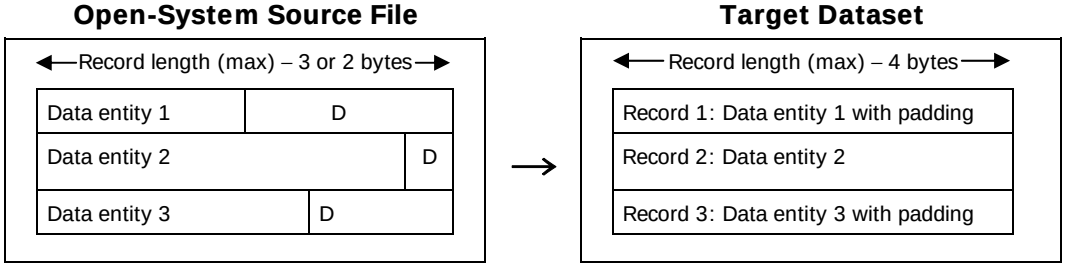


Figure 2-18 FXotm with Fixed-Length Records: Padding and Delimiters

Note: FCU does not extract padding from FXotm source files. If the FXotm source file contains delimiters but you specify **No** for the delimiter option, the delimiters will be regarded as part of the data entities and will be transferred to the target dataset.

FXotm with Variable-Length Record Format

FXotm operations can be performed on variable-length source files only if delimiters have already been added to the source file (e.g., from a previous FXmto operation). If a variable-length source file without delimiters is processed, FCU will use the maximum record length to construct the target data entities, thereby corrupting the data and rendering the dataset unusable. FCU extracts but does not add delimiters to FXotm source files.

With delimiters. The figure below shows an FXotm operation for a variable-length source file with delimiters. FCU extracts and transfers the data entities to the target dataset, and automatically adds the four-byte RL field. The delimiters are not transferred. The target dataset must have variable-length record format.

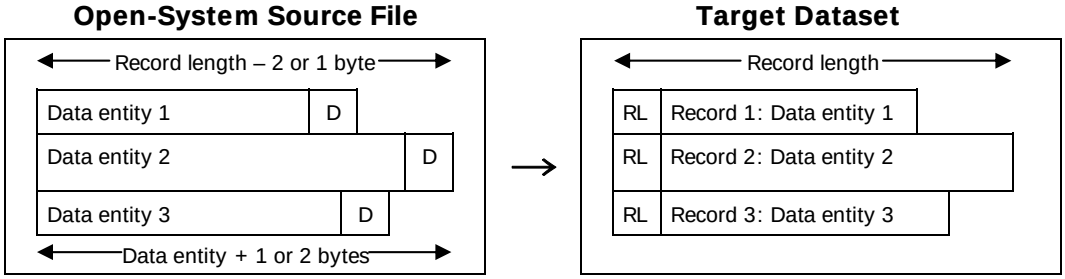


Figure 2-19 FXotm with Variable-Length Records: Delimiters

If the length of any data entity in a UNIX source file is greater than the maximum record length minus one byte (CR or LF delimiter), FCU aborts the operation and reports an error. If the length of any data entity in a Windows 2000/2003/Windows NT source file is greater than the maximum record length minus two bytes (CR+LF delimiter), FCU aborts the operation and reports an error.

With padding and delimiters. The figure below shows an FXotm operation with padding for a variable-length source file with delimiters. FCU version 01-01-41 or later is required for FXotm with padding added to variable-length source files. FCU adds padding, extracts and transfers the data entities with padding to the target dataset, and automatically adds the four-byte RL field. The delimiters are not transferred. The target dataset must have fixed-length record format with record length defined as needed.

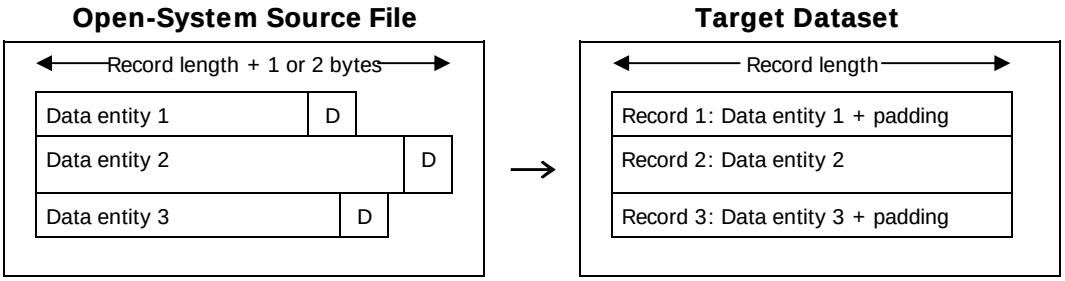


Figure 2-20 FXotm with Variable-Length Records: Padding and Delimiters

If the length of any data entity in a UNIX source file is greater than the specified record length plus one byte (CR or LF delimiter), FCU aborts the operation and reports an error. If the length of any data entity in a Windows 2000/2003/Windows NT source file is greater than the specified record length plus two bytes (CR+LF delimiter), FCU aborts the operation and reports an error.

FXoto Operations

FXoto operations transfer data from source files on one open-system platform to target files on another open-system platform. Each FXoto file transfer consists of two separate FX operations: first an FXotm operation transfers the data in the source file to an intermediate dataset, and then an FXmto operation transfers the data from the intermediate dataset to the target file. For any users with the all-open TagmaStore USP or NSC subsystem (no attached z/OS host), the intermediate datasets are allocated on OPEN-*x* FMT volumes. The FMT utility enables you to format OPEN-*x* LUs (standard or custom size) as FXoto volumes. The ALC utility enables you to allocate intermediate datasets on the OPEN-*x* FMT volumes. For users with the multiplatform TagmaStore USP or NSC subsystem, the intermediate datasets can be allocated on OPEN-*x* FMT volumes or on FX -A volumes, as desired. When you perform FXoto operations which access OPEN-*x* FMT volumes, the FXoto volume definition file must be available for use by FCU.

Note: FCU version 01-01-41 or later is required for FXoto operations using ALC-generated datasets on OPEN-*x* FMT volumes.

The FCU file transfer options (code conversion, padding, delimiters, etc.) can be used on the FXotm and FXmto sub-operations as needed.

- Code conversion is not available for FXoto transfers.
- Padding can be used but will render the target file incompatible with the source file due to the change in record format from variable-length to fixed-length. If you use padding for the FXotm operation, the target file can be transferred back to the same intermediate dataset but not back to the same source file. If you use padding for the FXmto operation, the target file cannot be transferred back to the same intermediate dataset or back to the same source file.
- Delimiters can be used to enable bidirectional data transfers. When using delimiters, watch out for files which contain the same character(s) as the delimiter (CR and/or LF) but used for purposes other than delimiting data entities. If you specify the delimiter option for FXotm, FCU will interpret all occurrences of the specified delimiter character(s) as delimiters, which can create a dataset with corrupt records or generate an error condition.
- The empty file option can be used to enable empty files to be processed. For example, if a source file specified in your FXoto FCU parameter definition file becomes empty, you can add the empty file option to the FXotm/mto operations on that file to enable FCU to process the FCU parameter definition file without errors.
- The RDW option is not normally used for FXoto operations. If you use the RDW option (FXmto operation only), you will not be able to transfer the data back to the same intermediate dataset.
- The VSE record option does not apply to FXoto operations which access ALC-generated intermediate datasets on OPEN-*x* FMT volumes. The only time you would use the VSE option is when transferring a file between open-system platforms via a VSE dataset on a -A FX volume. In this case, you must use the VSE record option for both transfers (FXotm/mto).

Host Access and I/O Contention

The user must manage access to the FXmto and FXotm volumes to prevent illegal I/O access contention between the z/OS and open-system hosts. These FX volumes cannot be accessed concurrently by the z/OS and open-system hosts, and must be varied offline from the z/OS host during FX operations. The FX volumes should not contain any regularly accessed data and should be dedicated to data exchange operations to avoid accidental overwriting of data.

Note: For 01-02-48 and earlier, do not access the FX volume from FAL/FCU when AIX is accessing it. For 01-02-50 and earlier, do not access the FX volume from FAL/FCU when Windows 2000/2003/Windows NT is accessing it.

Note: Please note the following restrictions for the listed operating systems:

- For UNIX operating systems, if the version is 01-02-48 and later, FXotm can run several different datasets simultaneously.
- For Windows 2000/2003/Windows NT operating systems, if the version is 01-02-50 and later, FXotm can run several different datasets simultaneously.
- For AIX operating systems, since volumes are reserved during accessing, FXotm cannot run several different datasets simultaneously.

Please refer to the *Reference Manual for RAID 200/300/400/450 FAL & FCU*, revision 20.0, for more detail.

The FX volumes can only be accessed by open-system hosts using the FAL/FCU software. The z/OS hosts have normal read/write access to the -B and -A volumes, read-only access to the -C volumes, and no access at all to the OPEN-*x* FMT volumes. The open-system hosts have read/write access to the -C, -A, and OPEN-*x* FMT volumes and read-only access to the -B volumes. The open-system hosts must use FAL/FCU to access all FX volumes.

WARNING: Concurrent access to the FX volumes by the z/OS and open-system hosts is not supported. The user is responsible for managing access to FX volumes to avoid I/O contention between the z/OS and open-system hosts. Since FCU accesses only the VTOC area of the FX -B volumes, catalog or security control functions cannot be used to provide access control for the 3390-3B volumes.

The z/OS host can issue a **reserve** command to reserve a volume for exclusive use. The z/OS **reserve** command prevents access by all other hosts, including all other z/OS hosts and all open-system hosts. The open-system host can also reserve a volume to exclude I/Os issued by other systems. The open-system **reserve** command prevents access by all other open-system hosts, but z/OS hosts still have normal access to FXmto and FXotm volumes reserved by open-system hosts. These **reserve** commands affect FX operations as follows:

- **Reserved by z/OS host**. When an FX volume is reserved by the z/OS host, FX operations cannot be performed on that volume, because the FAL/FCU access from the open-system host will terminate unsuccessfully. Open-system access other than read or write I/Os can be executed successfully.

  Note: Open-system access to an z/OS-reserved volume may complete successfully if the open-system retries the operation after the reserve is released. However, since the time interval before a retry varies depending on the open-system platform and the z/OS application that issued the reserve, the success of retry operations on reserved volumes cannot be guaranteed.

- **Reserved by open-system host**. When an FX volume is reserved by the open-system host, FX operations can be performed only from the host that reserved the volume. FX operations from any other open-system host will terminate unsuccessfully. Open-system reserve does not affect z/OS access to the FX volume.
- **Unreserved**. When an FX volume is not reserved by any z/OS or open-system host, FX operations can be performed from any open-system host using FAL/FCU. All z/OS hosts and all open-system hosts have access to unreserved volumes.

The user should implement exclusive access control and job coordination at the system level for the FX volumes. The user should also take the following steps to avoid I/O contention problems for the FX volumes:

- **Open-system access**. When the open-system host needs to access an FX volume, vary the volume and its channel path offline from all z/OS hosts.
- **z/OS access**. When the z/OS host needs to access an FX volume, stop all open-system access to the corresponding LU. For AIX, vary off the volume group(s). For Windows 2000/2003/Windows NT, use **unaccess**. Do not use any open-system program which accesses unmounted LUs (e.g., AIX SMIT, HP-UX SAM, and NT Disk Administrator).

AIX Shared Open Function

To share FileExchange volumes on multiple AIX operating systems:

- For FX versions 01-xx-59 and earlier, when one AIX OS opens a FileExchange volume, the other AIX OS cannot open the FileExchange volume. This is because the AIX OS reserves the FileExchange volume when it opens the it.
- For FX 01-xx-60 and later, it is possible to share an FileExchange volume across multiple AIX operating systems by specifying the environment variable:

```
FAL_NO_RESERVE.
```

The following table (Table 2-6) shows the relationship between shared volumes and FX versions.

Table 2-6 Shared Volume and FX Version

Object	Version	Description
FAL for AIX (32/64 bit)	01-xx-59 and earlier	Cannot open a shared volume from multiple AIX OSs.
	01-xx-60 and later	Can open a shared volume from multiple AIX OSs by specifying the environment variable.

To define environment variables:

- By specifying the environment variable (FAL_NO_RESERVE), you can select **shared open** or **exclusive open**. Table 2-7 shows the relationship between the environment variable (FAL_NO_RESERVE) and open mode.

Table 2-7 Environment Variable (FAL_NO_RESERVE) and Open Mode

FAL_NO_RESERVE	Open Mode
No definition of environment variable	Exclusive Open (Original mode)
ON	Shared Open
OFF	Exclusive Open
Other	Exclusive Open

AIX Reserve Function

AIX Reserved Retry Function

For FX versions 01-*XX*-61 and earlier, an AIXsystem error will occur when you use FX to send data to a reserved volume on another system. For FX versions 01-*XX*-62 and later, you can retry sending data to the reserved volume by specifying the wait time and retry count for the environment variable of the other system. FX can send data again when the reserved volume is released by the other system. The following table shows the AIX reserve functions.

Table 2-8 AIX Reserve Retry Function

Environment Variable or Function	Description
FAL_RETRY	To set retry function: specify the environment variable.
FAL_RETRY_COUNT	To set retry count: specify the environment variable. (FX attempts top send data while in reserve status.)
FAL_RETRY_WAIT_TIME	To set the interval: specify the environment variable. (FX attempts to send data while in reserved status.)
FAL_RETRY_TARGET	To set system error codes: Specify the environment variable. (System error codes asreretry triggers.)
Retry function	The retry function in invoked when AIX commands trigger a system error (open,close,read,write,seek,flush).
Out put retry log function	FX outputs a retry log (FAL_Error log) when it executes retry.

Environment Variables and Functions

- **Retry Function:** You can set the retry function by specifying the environment variable (FAL_RETRY).
  - If there is no definition of environment variable, the Retry function is **disabled**.
  - If the environment variable is defined as **ON**, the Retry function is **enabled**.
  - If the environment variable is defined as **OFF**, the Retry function is **disabled**.
  - If the environment variable is defined as **other**, the Retry function is **disabled**.
- **Retry Counts Function:** You can set the number of retry function instances by specifying the environment variable (FAL_RETRY_COUNT). FX attempts to send data during reserved status.
  - The default count number is 10, where no definition of the environment variable is given.
  - The range of the FAL_RETRY_COUNT variable is 1-600.

  - If other values outside the range are given, the default value applies.

- **Retry Wait Time Function:**You can set the wait time interval by specifying the environment variable(FAL_RETRY_WAIT_TIME). FX attempts to send data during reserved status.
  - The default is 1 second, where no definition of the environment variable is given.
  - The range of the FAL_RETRY_WAIT_TIME variable is 1-60 seconds.
  - If other values outside the range are given, the default value applies.
- **Retry target function:** You can set system error codes, which are retry triggers, by specifying the environment variable (FAL_RETRY_TARGET).
  - You can set up to 5 system error codes separated by commas. If you set six or more system error codes, FX will ignore all after the fifth.
  - If no environment variable definition is given, an error code 16(EBUSY) will appear.
  - You can set up to 5 system error codes, separated by commas.
  - If other values outside the range are given, the 16(EBUSY) error code applies.
  - For setting error codes outside default values, please contact your Hitachi Data Systems technical representative.

Bidirectional Data Transfer

FX supports bidirectional data transfer for both fixed-length and variable-length z/OS datasets. Bidirectional data transfer involves transferring data from z/OS datasets to open-system files and then back to the original z/OS datasets again. The requirements for bidirectional data transfer are:

- For all FXmto operations, do not specify the record description word (RDW) option. If the RDW option is specified for an FXmto data transfer, the subsequent FXotm target dataset will not be compatible with the original dataset.
- For FXmto with fixed-length datasets, do not specify the delimiter option, since the data entities are extracted by length. If you add delimiters for the FXmto transfer, the subsequent FXotm target dataset will not be compatible with the original dataset.
- For FXmto with variable-length datasets, you must add delimiters but not padding. If delimiters are not added or if padding is added for the FXmto transfer, the subsequent FXotm target dataset will not be compatible with the original dataset.
- For FXotm operations do not specify the delimiter option if the source file contains the same character(s) as the delimiter (CR and/or LF) but used for purposes other than delimiting data entities. If you specify the delimiter option for FXotm, FCU will interpret all occurrences of the specified delimiter character(s) as delimiters, which can create a dataset with corrupt records or generate an error condition.

3

Preparing for Cross-OS File Exchange Operations

This chapter describes the system requirements and volume configuration necessary to run and operate FX. It also provides instructions on how to install FX.

- ☐ System Requirements
- ☐ Installing and Configuring the FX Volumes
- ☐ Installing the FAL/FCU Software

System Requirements

The user should examine existing data exchange needs carefully, especially the desired number of FX volumes to be installed and configured, prior to TagmaStore USP or NSC system configuration. This is due to the possible need to reconfigure and reformat entire array groups, depending on the microcode level of the subsystem.

The system requirements for FX are:

- FAL/FCU CD for the applicable open-system platform(s). This revision of the *FX User's Guide* covers versions 01-02-4*x* through 01-03-59. For earlier versions, please refer to the previous revision of this document.
  - FAL/FCU currently supports files larger than 2 GB.
- Multiplatform or all-open TagmaStore USP or NSC subsystem:
  - FXmto: the TagmaStore USP or NSC must be configured with -B and/or -A FX volumes.
  - FXotm: the TagmaStore USP or NSC must be configured with -C and/or -A FX volumes.
  - FXoto: the user can format OPEN-*x* LUs as FXoto volumes or use -C FX volumes. Version 01-01-41 or later of FCU is required to format FXoto volumes.
- Remote Console PC and LUN Manager Software. LUN Manager enables the user to configure FC ports and create custom-size LUs. If remote LUN Manager is not installed, please contact the Hitachi Data Systems representative about LUN Manager configuration services. For information on LUN Manager, refer to the *Hitachi TagmaStore USP and NSC LUN Manager User's Guide* (MK-94RD203).
- z/OS operating systems: MVS, VSE (supported by FCU 01-01-40 and later), VOS3
  - Both 3990 and 2105 control unit (CU) emulations are supported.

- Open-system platforms and operating system (OS) version level(s):
  - 32bitFAL:

    HP-UX 10.2 and 11.0, 11i, 11iV2
    Solaris 2.5, 2.6, 7, 8, 9 (version 2.5 = Ver. 01-XX-47 or earlier)
    IBM® AIX® 5.3, 5.2ML02, 5.1ML02, 4.1.x, 4.2.x, 4.3.xML02, 5 (version 4.1 = Ver. 01-XX-47 or earlier)
    Windows® 2000 SP4(x86)/2003 SP1(x86)/NT® 4.0 SP6(x86)(IA64)(x64) (Workstation or Server™)
    HP® Tru64 UNIX V5.1a

    Red Hat Linux 7.2: AS2.1, AS3.0/IA64
  - 64bitFAL:

    HP-UX 11.0, 11i, 11iV2

    Solaris 8, 9

    IBM AIX 5.3, 5.2, 5.1, 4.3.3

    Red Hat Linux AS3.0/Update3(x86) or Update4/IA64

    Red Hat SuSE SLES9(x86)(IA64)(AMD64/EM64T)

Note: When installing/uninstalling Rapid Xchange (for 32/64bit), follow the procedure presented in this Chapter. For 64bit installation, the FX program must be version 01-03-59 or later.

  - 64bitFCU:

    HP-UX11.0,11i,11iV2

    Solaris 7,8,9

    IBM AIX 4.3, 5L

    Red Hat Linux AS3.0/IA64

    Note: The Motif 1.2 (or later) Windows system software is required for the FCU GUI for UNIX. If Motif is not installed, see Appendix B (*Using FCU without the GUI*).
  - Superuser (root) login access to the open-system server/workstation is required.

Note: For FAL/FCU versions earlier than 01-02-50, Multiple Volume Dataset is not supported. When a dataset is spanned over multiple volumes, only the portion of the dataset within one volume will be processed. For FAL/FCU versions 01-02-50 or later, Multiple Volume Dataset is supported only for FXmto.

Note: For 64bitFAL, set the HBA which supports 64bit to the server. IBM AIX systems support 64bitFAL. For further information on IBM AIX systems, please contact the IBM Support Center.

64-bit Version FCU

FX supports the 64-bit version FCU. The application method is the same as for the 32-bit version FCU.

Note: The 64-bit version FCU does not support a GUI interface.

FX version 01-04-61/20 supports Solaris, HP-UX, AIX 4.3, and AIX 5L operating systems.

FX version 01-04-62/20 supports Red Hat Linux AS3.0/IA64 operating system.

VSE Requirements and Restrictions

FX supports the VSE operating system. The following sections describe the supported VSE versions that correspond to the TagmaStore subsystem and associated FX version.

Note: There is no guarantee that FX will correctly access your data without meeting these requirements.

Table 3-1 Support and OS Conditions

OS version	RAID subsystem	FX
VSE2.5 and upper	TagmaStore USP or NSC	After 01-03-59

When you use MTO and OTM for the dataset allocated by VSE2.3, you must specify RF (Record Format), RL(Record Length) and BL(Block Length) by using the correct VSE parameters.

When you use MTO and OTM for the dataset allocated by VSE2.5, FX can transfer data without a VSE parameter.

Table 3-2 Support Matrix for VSE OS, VSE Parameter and Record Format

OS andFX Version	VSE Parameter	Record Format			
		F	FB	V	VB
VSE2.3 and previous	Yes				
(FX: 01-xx-58 and previous)	No	×	×	×	×
VSE2.5 and higher	Yes			Note 1	Note 2
(FX: 01-03-59)	No			Note 3	Note 4

Key:

o: Transfer is available
×: Transfer is not available
F: Fixed non-block length
FB: Fixed block length
V: Variable non-block length
VB: Variable block length

Note 1: It is possible to transfer data between correct dataset attribute(5≦RL≦BL-4). The data transfer is valid only if the VSE parameters are as shown:

RL≦32756

BL≦32760

BL=RL+4

For the following dataset attributes, the data transfer is invalid if user does **not** specify the VSE parameter value as shown above.

RL>32756

BL>32760

For the following dataset attributes, the data transfer is invalid if user does **not** specify the VSE parameter value between RL and BL values shown in #1 and #2.

RL≦32756

BL≦32760

RL=BL

s#1: RL(Input value for VSE parameter) = RL(value on VTOC) + 4 ≦32756

#2: BL(Input value for VSE parameter) = BL(value on VTOC) + 8 ≦32760

Note 2: It is possible to transfer data between the correct dataset attributes (5≦RL≦BL-4). The data transfer is invalid if the VSE parameter is **not** the following value:

RL(Input value for VSE parameter) = RL(value on VTOC) + 4 ≦32756

BL(Input value for VSE parameter) = BL(value on VTOC) + 8 ≦32760

Note 3: It is possible to transfer data between the correct dataset attributes (BL=RL+4≦32760). The data transfer is invalid if the RL and BL values on VTOC do **not** match to the following condition:

BL=RL+4≦32760

When the RL and BL values on VTOC are such that RL > 32756 and BL > 32760, FX manages the data as RL=32756 and BL=32760.

When the RL and BL values on VTOC are such that RL≦32756, BL≦32760 and RL = BL, FX manage the data as shown below:

RL(FX internal value) = RL(value on VTOC) + 4 ≦32756

BL(FX internal value) = BL(value on VTOC) + 8 ≦32760

Note 4: It is possible to transfer data between the correct dataset attributes (BL=RL+4≦32760). The data transfer is invalid if RL and BL values on VTOC do **not** match to the following condition:

BL=RL+4≦32760

When the RL and BL values on VTOC are such that RL=BL≦32752, FX manages the data as shown below:

RL(FX internal value) = RL(value on VTOC) + 4

BL(FX internal value) = BL(value on VTOC) + 8

Note: If you Create a user program by using the dataset attribute exchange rule, you need to use the following functions:

datasetGetFileInformation()

datasetGetFileInformationEx()

Compiler Requirements

The compiler requirements are shown below for each OS. This information is the result of test and evaluation by Hitachi LTD and is guaranteed for the specified development environment for use with FX. If the development environment differs from that specified here, your results may differ.

Table 3-3 Platforms and Associated Operating Systems

Operating System / Platform	Compiler
Red Hat Linux 7.2 (Kernel version 2.4.7-10) and Red Hat Linux AS2.1 and AS3.0	gcc(Ver.2.96-98) glibc(Ver.2.2.4-13)
Solaris	Work Shop Compilers 4.2.1
Windows NT 4.0 service Pack 6a	Visual C++® 6.0 SP4
HP-UX 10.20	HP® 92453-01 A.10.32.03 HP® C Compiler
AIX4.3	C® for AIX 4.4
AIX5L	C® for AIX, Version 5.0
HP® Tru64	Digital UNIX Compiler Driver 3.11 DEC C V5.6-075 on Digital UNIX V4.0 (Rev. 878)

Maximum Data Size

Table 3-4 describes the maximum possible data sizes for FAL/FCU version 01-02-48 or later.

Table 3-4 Maximum Data Size

Emulation Type	No LUSE	LUSE	Multiple Volume Dataset
OPEN-3	About 21 GB	About 42.8 GB	
OPEN-8	About 6.5 GB	About 42.8 GB	
OPEN-9	About 6.5 GB	About 42.8 GB	
OPEN-E	About 12.9 GB	About 42.8 GB	
OPEN-L	About 32.3 GB	About 42.8 GB	
OPEN-V	About 42.8 GB	About 42.8 GB	
3390-3X	About 2.6 GB		OS limits: HP-UX, Solaris, NT, Tru64®: 78GB AIX: 64GB
3390-9X	About 8.3 GB		HP-UX(*1)/Solaris /NT Tru64 :249GB AIX: 64GB(The limit of OS)
3390-LX	About 27.1 GB		HP-UX(*1)/Solaris /NT/ Tru64 :813GB AIX: 64GB(The limit of OS)

Note: The data capacity that can be stored within the intermediate file is smaller than its physical capacity, and varies depending upon the block length to be used.

Additional Notes:

- 3390-9A: Can be used for both FileExchangemto and FileExchangeotm. The same access as for 3390-9 is allowed from Mainframe hosts. Read and write are possible from Open system hosts.
- 3390-9B: Can be used only for FileExchangemto. The same access as for 3390-9 is allowed from Mainframe hosts. Read only is allowed from Open system hosts.
- 3390-9C: Can be used only for FileExchangeotm. The same access as for 3390-9 is allowed but read only from Mainframe hosts. Read and write are possible from Open system hosts.
- 3390-LA: Can be used for both FileExchangemto and FileExchangeotm. The same access as for 3390-L is allowed from Mainframe hosts. Read and write are possible from Open system hosts.
- 3390-LB: Can be used only for FileExchangemto. The same access as for 3390-L is allowed from Mainframe hosts. Read only is allowed from Open system hosts.
- 3390-LC: Can be used only for FileExchangeotm. The same access as for 3390-L is allowed but read only from Mainframe hosts. Read and write are possible from Open system hosts.

Interoperability with HDLM

When FX is used with HDLM, the FAL/FCU version must be 01-03-56 or later. The appropriate support platform versions are shown in Table 3-5. Please refer to the *Hitachi Freedom Storage Hitachi Dynamic Link Manager (HDLM) User's Guide for IBM AIX Systems* (MP-92DLM111), the *Hitachi Freedom Storage Hitachi Dynamic Link Manager (HDLM) User's Guide for HP-UX Systems* (MP-92DLM112), the *Hitachi Freedom Storage Hitachi Dynamic Link Manager (HDLM) User's Guide for Sun Solaris Systems* (MP-92DLM114), and the *Hitachi Freedom Storage Hitachi Dynamic Link Manager (HDLM) User's Guide for Windows NT and Windows 2000* (MP-92DLM129) for further details on each platform.

Table 3-5 HDLM Platforms and HDLM Versions

Platform	HDLM Version
Windows 2000/NT/2003	04-01, 05-00-/B
AIX	05-00-/A, 05-01
HP-UX	04-00-/A
Solaris	04-01, 05-00

Installing and Configuring the FX Volumes

The FX volumes are installed and configured during TagmaStore USP and NSC installation and configuration. The FX volumes should be dedicated to data exchange operations to avoid accidental overwriting or deletion of important data. The FXmto volumes (-B and -A) contain z/OS data to be transferred to open-system LUs. The FXotm volumes (-C and -A) contain open-system data to be transferred to z/OS volumes. The FXoto volumes contain the intermediate datasets for file transfers between open-system platforms. FX does not support concurrent access to FX volumes by the z/OS and open-system hosts.

To install and configure the FX volumes:

1. Determine exactly how many FXmto, FXotm, and FXoto volumes you will need for your multiplatform data exchange operations. The -A volumes can be used for FXmto, FXotm, and FXoto. The -B volumes are restricted to FXmto. The -C volumes are restricted to FXotm. The OPEN-*x* FMT volumes are restricted to FXoto. Make sure that the Hitachi Data Systems representative installs the desired number of each type of FX volume during TagmaStore USP and NSC subsystem installation and configuration.

   Note: If you need to change the number of FX volumes, please contact your Hitachi Data Systems account team. Reconfiguring the FX volumes after TagmaStore USP and NSC subsystem installation may require reformatting entire array groups, depending on the microcode level of the subsystem.

2. Complete TagmaStore USP and NSC installation and device configuration as specified in the TagmaStore USP and NSC configuration guide for the applicable open-system platform (e.g., the *TagmaStore USP and NSC HP-UX Configuration Guide*, MK-94RD235).
   - **Device recognition and device files**. For all open-system platforms, you must verify device recognition and device file creation for all FX volumes.
   - **File system / volume group**. Do not create a file system or volume group on any FX volume, including the OPEN-*x* devices which will be formatted for FXoto operations. FX volumes can only be accessed as raw devices by the open-system host using FAL/FCU (no mount operation).

– **Defining RAW Devices.** Those volumes which are to be used as intermediate volumes and to be shared between open systems must be defined as **OPEN-3/8/9/K/E/L/M/V** emulation type for RAID200/300, and must be defined as "raw" devices from each host server. From the open systems, there are no means to distinguish OPEN-3/8/9/K/E/L/M/V for open system dedicated volumes from these FX volumes. Please make sure not to confuse the usage on those volumes in the host systems.

  - The operations below which create file systems on the intermediate volumes must not be executed. Otherwise, information on the volume may be destroyed and the volumes will become unusable as FX volumes.
    – Solaris: "**newfs**" command
    – HP-UX: "**pvcreate**" command
    – IBM AIX: creating a volume group
    – Windows: formatting and creating a file system
    – Digital UNIX/HP® Tru64 UNIX: "**newfs**" command
    – SequeNT DYNIX/ptx®: creating a file system
    – NCR® SVR4: creating a volume group
    – Linux: "**raw**" command

    Note for Microsoft®: Cluster Server: When installing FX devices in a Microsoft Cluster Server (MSCS) environment, you must write signatures on the FX volumes before configuring MSCS.

  – The MSCS server cannot connect volumes which do not have signatures.
  – The volume on which a signature is written cannot be accessed from another server.
  – The volume on which a signature is written cannot be shared.
  – Only the mainframe and the server which wrote the signature can access the volume which has the signature.

– **I/O time-out and I/O queue depth**. Make sure to set the I/O time-out and I/O queue depth values for the FX volumes as specified in the applicable TagmaStore USP and NSC configuration guide.

– **Partition size**. Make sure to specify the correct partition size for the FX volumes as specified in the applicable TagmaStore USP and NSC configuration guide. If the partition size for -A or -B volumes is smaller than the mainframe volume size, the open-system host may not be able to access data to the end of the extent of these volumes.

For Solaris, use the following partition sizes for the FX volumes, and use 2 (two) for the number of alternate cylinders (Table 3-6):

Table 3-6 Partition Sizes for TagmaStore USP and NSC Subsystems

LVI	Cylinder # for Data Cylinder Extent
3390-3A	0 - 3345
3390-3B	0 - 3339
3390-3C	0 - 3345
3390-9A	0 - 10035
3390-9B	0 - 10017
3390-9C	0 - 10035
3390-LA	0 - 32763
3390-LB	0 - 32760
3390-LC	0 - 32763
OPEN-3	0 - 3335
OPEN-8	0 - 9963
OPEN-9	0 - 10013
OPEN-E	0 - 19756
OPEN-L	0 - 49433
3390-3A	0 - 3345
3390-3B	0 - 3339
3390-3C	0 - 3345
OPEN-3	0 - 3335
OPEN-8	0 - 9963
OPEN-9	0 - 10013
OPEN-E	0 - 19756
OPEN-L	0 - 19012
OPEN-V	See ***Note 1***.
3390-3A	0 - 3345

Note 1: For further information on Solaris cylinder partition sizes, please refer to the *Hitachi TagmaStore USP and NSC LUN Expansion and Virtual LVI/LUN User's Guide (*MK-94RD205), and the *Hitachi TagmaStore USP and NSC Sun Solaris Configuration Guide*(MK-94RD236)

- **Volume labels**. An FX volume with a volume label cannot be shared between open-system platforms which use volume labels. Table 3-6 shows the allowable configurations for sharing FX volumes between open-system platforms for the TagmaStore USP and NSC subsystems. Table 3-7 shows the allowable configurations for sharing FX volumes between open-system platforms for the TagmaStore USP and NSC subsystems. HP-UX and IBM AIX do not use volume labels, so FX volumes can always be shared with these platforms. Labels are optional for DIGITAL UNIX/Tru64 and Windows 2000/2003/Windows NT, so FX volumes can be shared with these platforms only if they have no label. Solaris always writes volume labels, so FX volumes can never be shared between these two platforms, but can be shared with the other platforms (HP, IBM, DIGITAL UNIX/Tru64, NT) as long as they do not have labels.

  Note: Solaris may display the following warning messages when formatting and labeling an FX volume. This is normal, and the user can ignore these messages.

  Warning: error writing VTOC
  Warning: no backup labels
  Write label failed

3. For UNIX hosts, make sure to set up the desired access privileges for each FX volume (e.g., using groups and/or chmod command). Please refer to the OS user documentation for information on access permission rights. For Windows 2000/2003/Windows NT, Administrator access is required to access the FX volumes.
4. On the z/OS host, make sure to initialize and write the VTOC for each FXmto and FXotm volume to enable the z/OS host to access the volumes. The ICKDSF media maintenance utility can be used to perform these tasks.
5. After FAL/FCU software installation, make sure to format each FXoto volume using the FX Formatter (FMT) utility on the UNIX/ Windows NT host. This enables the FXoto intermediate datasets to be allocated.

Table 3-7 Sharing FX Volumes between Open-System Platforms

		IBM AIX	HP-UX	SequeNT DYNIX®	Windows 2000/2003/ Windows NT	HP®True64 ® UNIX®	Solaris	NCR® SVR4	Linux
No label	IBM AIX	OK	OK	OK	OK	OK	OK	OK	OK
	HP-UX	OK	OK	OK	OK	OK	OK	OK	OK
	SequeNT DYNIX	OK	OK	OK	OK	OK	OK	OK	OK
	Linux	OK	OK	OK	OK	OK	OK	OK	OK
Label write option	HPTrue64 / UNIX	OK	OK		OK	CHK	CHK		
	Windows 2000/2003/ Windows NT	OK	OK	OK	OK	CHK	CHK	CHK	OK
Label auto-write	Solaris	OK	OK	OK	OK	OK	OK		
	NCR SVR4	OK	OK	OK	OK	OK	NO	NO	OK

CHK = sharing allowed only if volume has no label.

Installing the FAL/FCU Software

The FAL/FCU software must be installed on the UNIX/NT server(s) attached to the subsystem. FAL/FCU software installation for the UNIX-based platforms is different from FAL/FCU installation on Windows 2000/2003/Windows NT. The following sections provide instructions for installing FAL/FCU version 01-01-36/38 or later. For instructions on installing versions 01-01-24 and earlier, please refer to previous revisions of this document.

Installing FAL/FCU on UNIX-Based Platforms

32-Bit FAL/FCU Software

To install the 32-bit FAL/FCU software on a UNIX-based platform:

1. If FAL/FCU version 01-01-36 or later is already installed (by **cpio** command), you do not need to uninstall it. A new installation will overwrite the previous version.
   If FAL/FCU version 01-01-24 or earlier is already installed (by setup program), you must uninstall this older version as follows before installing version 01-01-36 or later:
   a) Log in with the same user ID that was used to install the old FAL/FCU software.
   b) If FAL/FCU was installed from FD using the setup program, see **Uninstallation** to remove FAL/FCU.
   c) If FAL/FCU was installed from DAT, or if you can't find your FAL/FCU installation FD, move to the directory **fcu/fal.o/dataset.h** and remove FAL/FCU by entering:
      # rm fcu fal.o dataset.h $HOME/FcuMf

      If you cannot find the directory, you can use the following procedure:
      # find / -name "fcu" -print
      # find / -name "fal.o" -print
      # find / -name "dataset.h" -print
      # find / -name "FcuMf" -print
2. Log in to the system as **root**.
3. Insert the FAL/FCU installation media (e.g., CD-ROM) into the drive.

   Note: Verify that the device file for the CD-ROM drive exists. For Solaris, do not use **volcheck** if the CD-ROM device file is not available for auto-mount.

4. Make sure the following six directories exist on the open-system host. If not, create the directories using the **mkdir** command (e.g., **# mkdir /usr/lib/X11/app-defaults**).

/usr	/usr/lib
/usr/bin	/usr/lib/X11
/usr/include	/usr/lib/X11/app-defaults

5. Move to the **root** directory.
6. For UNIX-based systems, copy the FAL/FCU software from the installation CD-ROM as follows:

```
# cpio -iBmuv < CD_device_file_name/d
```

Note: Use the full device file name: wildcards will not work.

7. For Solaris you must set a path to the resource file for each FAL/FCU user:

   a) For C shell, add the following line to the end of the **.cshrc** file in the home directory. If **.cshrc** does not exist, create it and enter the following line:

      setenv XFILESEARCHPATH /usr/lib/X11/app-defaults/%N:$XFILESEARCHPATH

      export XFILESEARCHPATH

      Note: Add these two lines to the file ".profile" in your home directory, when it is not in the common desktop environment. If ".profile" does not exist, create it.

   b) For non-C shell, add the following two lines to the end of the **.dtprofile** file in the home directory. If **.dtprofile** does not exist, create it and enter the following lines:

      XFILESEARCHPATH=/usr/lib/X11/app-defaults/%N:$XFILESEARCHPATH
      export XFILESEARCHPATH

   c) You must log out and log back in to implement these changes.

8. For Tru64 UNIX you must uncompress the FAL/FCU program:

   # uncompress /usr/bin/fcu.Z

9. Remove the CD-ROM from the drive.

10. Log out, and then log in again.

Note: When the FX Code Converter is installed, the libuoc.* file is replaced with the FX Code Converter library (the extension varies according to OS). Before installing FX Code Converter, save libuoc.* with an alias.

64-Bit FAL/FCU Software

To install the 64bitFAL/FCU software on a UNIX-based platform:

1. Log-in as "root".
2. Set CD-ROM, in which the 64bitFAL/FCU is stored, to the drive.
3. Mount it.
4. Check to see if the following directories currently exist. If they do not, create them as follows:
   a. All Platforms: /usr, /usr/lib
   b. Solaris: /usr/lib/sparcv9
   c. HP-UX: /usr/lib/pa20_64
5. Move to the root directory
6. Copy 64bitFAL from CD-ROM
7. A file or directory can be viewed using the correct file name given at mounting. To view a directory, use one of the following procedures according to platform:
   d. Install 64bitFAL after confirming a directory name and a file name by using the **ls** command.

   – HP-UX (11.0):

     #cpio -iBmuv <(MountPoint)/PROGRAM/FAL64/HP_UX/HP_UX.CPI

   – Solaris (Solaris 7,8, 9):

     #cpio -iBmuv (MountPoint)/PROGRAM/FAL64/SOLARIS/SOLARIS.CPI

   – AIX (AIX 4.3.3):

     #cpio -iBmuv <(MountPoint)/PROGRAM/FAL64/AIX/AIX4/AIX.CPI

   – Red Hat Linux AS3.0/IA64:

     #cpio -iBmuv <(MountPoint)/PROGRAM/FAL64/LINUX/LINUX.CPI
8. Remove the CD-ROM from the drive.
9. Log-out once and log-in again

Installing FAL/FCU on Windows

To install the FAL/FCU software on a Windows 2000/2003/Windows NT-based platform:

1. If FAL/FCU is already installed, uninstall it before installing the new version.
   a) To uninstall FAL/FCU version 01-01-25 or later, use the Windows **Add/Remove Programs** utility.
   b) To uninstall FAL/FCU versions 01-01-24 and earlier, delete the folder that contains the FAL/FCU software components (**fcu.exe**, **fal.obj**, and **dataset.h**).
2. Insert the FAL/FCU installation CD-ROM into the drive, and run **setup.exe**.

Note: For Windows 2000/2003/Windows NT, if the **Installed Directory** has a directory name using a “space” character, enter the following: **<license key> fal.dll falmt.dll**

Uninstalling the FAL/FCU Software on UNIX-Based Platforms

If you need to uninstall FAL/FCU version 01-01-36 or later:

1. Log in to the system as **root**.

   Remove the FAL/FCU for 32bit files using the rm command as follows, or string the commands:

   ```
   # rm /usr/bin/fcu
   # rm /usr/bin/fcunw
   # rm /usr/include/dataset.h
   # rm /usr/lib/libfal.
   # rm /usr/lib/libuoc.
   # rm /usr/lib/X11/app-defaults/FcuMf
   # rm /usr/bin/mfformat
   # rm /usr/bin/allocds
   ```

2. Remove the FAL/FCU for 64bit files using the **rm** command as follows, or string the commands:

   ```
   # rm /usr/bin/fcunw
   # rm /usr/include/dataset.h
   # rm /usr/lib/libfal64.*
   # rm /usr/lib/libuoc64.*
   # rm /usr/bin/mfformat64
   # rm /usr/bin/allocds64
   ```

   When uninstalling 32bit FAL/FCU version 01-01-41 and later, remove the following file:

   # rm /usr/bin/listvol

   When the 64bit FAL/FCU Ver.01-01-41/00 or later is used, remove the following files:

   #rm /usr/bin/ppkeyset64

   #rm /usr/bin/autoppkeyset64

   When uninstalling 32bit FAL/FCU version 01-01-45 and later, remove the following files:

   # rm /usr/lib/libfal.ver

   #rm /usr/bin/ppkeyset

   #rm /usr/bin/autoppkeyset

Uninstalling FAL/FCU on Windows

To uninstall the FAL/FCU software on a Windows 2000/2003/Windows NT-based platform:

3. Depending on your version of FAL/FCU, do one of the following:

- To uninstall FAL/FCU version 01-01-25 or later, use the Windows **Add/Remove Programs** utility.
- To uninstall FAL/FCU versions 01-01-24 and earlier, delete the folder that contains the FAL/FCU software components (**fcu.exe**, **fal.obj**, and **dataset.h**).

Note: For Windows 2000/2003/Windows NT, if the **Installed Directory** has a directory name using a "space" character, enter the following: **<license key> fal.dll falmt.dll**

Entering the FX License Key Code

The license key for FX is entered by command on the server system after FX has been installed. A license key is required for each server and for each different server type. Each key is associated with a specific TagmaStore USP or NSC subsystem (defined by serial number). The procedures for entering license keys follow.

For UNIX platforms, input the following command from the command line:

- HP-UX:
  - 32bit: ppkeyset <License key> /usr/bin/fcunw /usr/bin/fcu /usr/lib/libfal.sl
  - 64bit: ppkeyset64 <License key> /usr/bin/fcunw64 /usr/lib/pa20_64/libfal64.sl
- Solaris:
  - 32bit: ppkeyset <License key> /usr/bin/fcunw /usr/bin/fcu /usr/lib/libfal.so.1
  - 64bit: ppkeyset64 <License key> /usr/bin/fcunw64 /usr/lib/sparcv9//libfal64.so.1
- AIX:
  - 32bit: ppkeyset <License key> /usr/bin/fcunw /usr/bin/fcu /usr/lib/libfal.a
  - 64bit: ppkeyset64 <License key> /usr/bin/fcunw64 /usr/lib/libfal64.a
- Digital UNIX/HP Tru64 UNIX /DYNIX/ptx:
  - 32bit: ppkeyset <License key> /usr/bin/fcunw /usr/bin/fcu /usr/lib/libfal.so
- Linux:
  - 32bit: ppkeyset <License key> /usr/bin/fcunw /usr/lib/libfal.so.1
  - 64bit: ppkeyset64 <License key> /usr/bin/fcunw64 /usr/lib/libfal64.so.1

Creating FXoto Volumes Using the FMT Utility

After the FAL/FCU software has been installed on the open-system host(s), you can format the FXoto volumes using the FX Formatter (FMT) utility. This enables you to allocate FXoto intermediate datasets. The FMT utility for UNIX is a UNIX command executed from the UNIX command line. The FMT utility for Windows 2000/2003/Windows NT is a GUI.

Note: FCU version 01-01-42 or later is required for the FMT utility.

The FX FMT utility defines the size of the OPEN-*x* volume in cylinders. The maximum number of cylinders allowed by FMT is shown in Table 3-8.

Table 3-8 FMT Utility Values

Emulation Type	01-XX-47 or earlier (*Note 2*)	01-XX-YY/ZZ not LUSE (*Note 2*)	01-XX-YY/ZZ LUSE (*Note 2*)	01-XX-YY/2x not LUSE (*Note 2*)	01-XX-YY/2x LUSE (*Note 2*) n=number of volumes (*Note 3*)
OPEN-3	3331	3331	5818	3331	(min 3338*n-7, 65534)
OPEN-8	5818	5818	5818	9959	(min 9996*n-7, 65534)
OPEN-9	5818	5818	5818	10009	(min 10016*n-7, 65534)
OPEN-E	not supported	5818	5818	19752	(min 19759*n-7, 65534)
OPEN-L	not supported	5818	5818	49429	(min 49439*n-7, 65534)
OPEN-V	not supported	not supported	not supported	65534	(min(Vc*n*128/ 96-7,65534) (see ***Note 4***)

Note 2: XX = 1 or 2; YY,x = number; ZZ<2*x*

Note 3: For Solaris, the data cylinder must be less than or equal to 32767. When using a LUSE volume, the geometry parameter is different, so the number of cylinders should be calculated as follows:

Cylinder (specified to FAL formatting) <= (A*B*C) / (15*96) – 5

A: Head (Geometry parameter)

B: Block/Track (Geometry parameter)

C: cylinder (Geometry parameter)

Note 4: Vc = OPEN-V cylinder value (MAX.:49160 cylinders using FAL)

The FMT utility can be used on standard-size OPEN-*x* volumes and on Virtual LVI/LUN (VIR) volumes.

Note: The VIR OPEN-*x* devices can also be called custom volume size (CVS) devices (e.g., OPEN-3-CVS). When formatting a VIR OPEN-*x* LU, use the number of cylinders defined for VIR minus seven (e.g., use 993 cylinders for a VIR device defined with 1000 cylinders). The cylinder size is: one cylinder = 15 tracks, one track = 96 sub-blocks, one sub-block = 512 bytes. Table 3-9 shows the relation between block length and write available capacity per track. The actual data capacity per cylinder = (write available capacity per track) × (15 tracks).

Table 3-9 Relation between Block Length and Write Available Capacity per Track

Block Length by Allocater = (A) (Bytes)	Write Available Data per Track (Bytes)	Block Length by Allocater = (A) (Bytes)	Write Available Data Per Track (Bytes)	Block Length by Allocater = (A) (Bytes)	Write Available Data Per Track (Bytes)
23477 - 32760	(A) × 1	1589 - 1684	(A) × 22	565 - 596	(A) × 44
15477 - 23476	(A) × 2	1493 - 1588	(A) × 23	533 - 564	(A) × 45
11477 - 15476	(A) × 3	1397 - 1492	(A) × 24	501 - 532	(A) × 46
9077 - 11476	(A) × 4	1333 - 1396	(A) × 25	469 - 500	(A) × 47
7477 - 9076	(A) × 5	1269 - 1332	(A) × 26	437 - 468	(A) × 48
6357 - 7476	(A) × 6	1205 - 1268	(A) × 27	405 - 436	(A) × 49
5493 - 6356	(A) × 7	1141 - 1204	(A) × 28	373 - 404	(A) × 50
4821 - 5492	(A) × 8	1077 - 1140	(A) × 29	341 - 372	(A) × 51
4277 - 4820	(A) × 9	1045 - 1076	(A) × 30	309 -340	(A) × 52
3861 – 4276	(A) × 10	981 - 1044	(A) × 31	277 - 308	(A) × 53
3477 –3860	(A) × 11	949 - 980	(A) × 32	245 - 276	(A) × 54
3189 – 3476	(A) × 12	917 - 948	(A) × 33	213 - 244	(A) × 55
2933 – 3188	(A) × 13	853 - 916	(A) × 34	181 - 212	(A) × 56
2677 – 2932	(A) × 14	821 - 852	(A) × 35	149 - 180	(A) × 57
2485 – 2676	(A) × 15	789 - 820	(A) × 36	117 - 148	(A) × 58
2325 – 2484	(A) × 16	757 - 788	(A) × 37	85 - 116	(A) × 59
2165 – 2324	(A) × 17	725 - 756	(A) × 38	53 - 84	(A) × 60
2005 – 2164	(A) × 18	693 - 724	(A) × 39	21 - 52	(A) × 61
1877 – 2004	(A) × 19	661 - 692	(A) × 40	1 - 20	(A) × 62
1781 – 1876	(A) × 20	629 - 660	(A) × 41	—	—
1685 – 1780	(A) × 21	597 - 628	(A) × 42	—	—

Note: The write available data per track includes the four-byte RL information and four-byte BL information for each record. When transferring variable-length records, make sure to take this extra required space into account.

WARNING: The FX FMT utility erases all data on the OPEN-*x* LU being formatted. If necessary, back up the data on the OPEN-*x* LUs prior to FMT formatting.

To format an OPEN-x volume using the FX FMT utility for UNIX:

1. Log in to the system as **root**.
2. Enter the following command at the UNIX command line prompt:
   # mfformat -d *devname* -v *VOLSER* [-p *primary_cylinders*]

   -d devname: Specify the raw device name (e.g., /dev/rdsk/c0t1d2 for HP-UX) of the OPEN-*x* volume being formatted. This parameter is required. Make sure to use the same raw device name for this volume in the FXoto volume definition file.

   -v VOLSER: Specify the VSN of the volume being formatted (A-Z, 0-9, @, #, \). Use only uppercase letters, and do not use any spaces or symbols other than @, #, and \. This parameter is required. Make sure to use the same volser for this volume in the FXoto volume definition file.

   -p primary_cylinders: Specify the number of primary cylinders (from decimal 2 through 5818). This parameter is required for custom-size volumes but is optional for standard-size volumes. If this parameter is omitted, the default value of max cylinders is used, specifically: OPEN-3 = 0-3331, OPEN-8 = 0-9959, OPEN-9 = 0-10009, OPEN-E = 0-19752, OPEN-L = 0-49429. When LUSE is set, the default value is the maximum value in a single volume.
3. If the FMT format operation could not be started due to an error condition, the **Format check error** message is displayed. If the FMT format operation did not complete successfully, an error message is displayed. Remove the error condition, and retry the operation (see section C.2 in Appendix C for further information on errors in UNIX).

To format an OPEN-x volume using the FX FMT utility for Windows 2000/2003/ Windows NT:

1. Log in to the system as administrator.
2. Double-click on the **Format** icon to open the Format panel (see Figure 3-1).
3. On the Format panel, enter the six-character volume serial number for the OPEN-*x* volume being formatted in the **VOLSER** field. Make sure to use the same volser for this volume in the FXoto volume definition file.
4. Specify the physical drive number (device number) for the OPEN-*x* volume being formatted in the **Physical drive No** field. Make sure to use the same physical drive number for this volume in the FXoto volume definition file.
5. Specify the number of cylinders for the OPEN-*x* volume in the **Cylinder Size** field. The **Min.** button enters 2 (two) cylinders, and the **Max.** button enters 5818 cylinders. If the OPEN-*x* volume is standard size (e.g., OPEN-3), use the maximum size of 5818 cylinders. If the OPEN-*x* volume is custom size (e.g., OPEN-3*n VIR device), use the following value: (# of cylinders defined for VIR) – 7. For example, if the VIR OPEN-*x* volume is defined with 1000 cylinders, enter 993 in the **Cylinder Size** field.

   Note: The maximum size for the Allocater is 4369 cylinders.

6. When the VSN, physical drive number, and cylinder size are correct, select the **Start** button. When the Format confirmation appears (see Figure 3-2), select **OK** to perform the requested FMT format operation, or select **Cancel** to cancel your request.
7. When the format operation completes successfully, the **Format complete** message is displayed (see Figure 3-3). If the specified volser has already been used, an error message is displayed to notify you (see Figure 3-4). If the format operation could not be started due to an error condition, the **Format check error** message is displayed. If the format operation did not complete successfully, one of the following error messages is displayed (see Figure 3-5) (n = system error code):

Open error! (n)	Open process error on specified volume.
Seek error! (n)	Seek process error on specified volume.
Read error! (n)	Read process error on specified volume.
Write error! (n)	Write process error on specified volume.
Close error! (n)	Close process error on specified volume.

8. When you are finished formatting OPEN-*x* volumes for use as FXoto volumes, select **Close** to close the Format panel and exit the FMT utility.

Important Note:

When the message "**Format check error**" is indicated, the formatting operation has not started and the original condition has been kept. When another message is indicated, the formatting process has already started. The data on the volume has already been initialized. Remove the error condition and format the volume again.

If you execute over the maximum number of cylinders on Windows 2003, the message "Format failed … Format check error" appears after about a minute. It should be clear when formatting has completed.

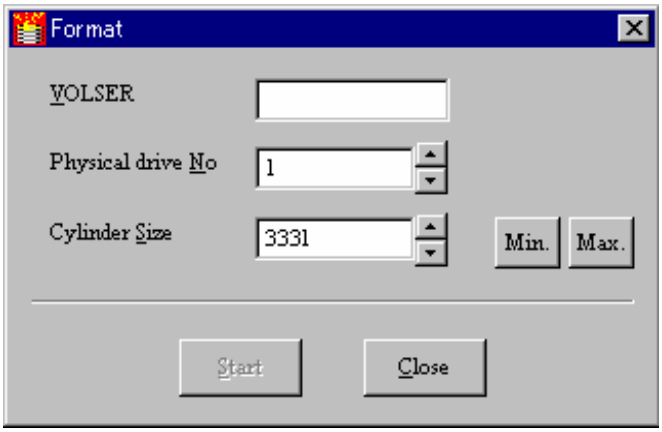


Figure 3-1 FMT Utility for Windows 2000/2003/Windows NT

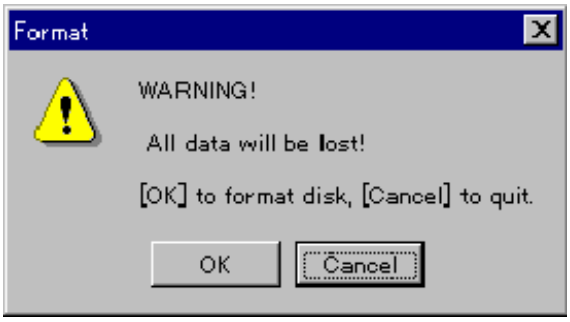


Figure 3-2 FMT Format Warning and Confirmation

Figure 3-3 FMT Format Complete Message

Figure 3-4 FMT VOLSER Used Message

Figure 3-5 FMT Error Message

Creating the FX Volume Definition File(s)

The FX volume definition file contains the volume association parameters for the FX volumes on the TagmaStore USP and NSC. This file must be created before you can use FCU or FAL to access data on these volumes. The volume association parameters define the FX volume by associating the volume serial number (VSN or volser) with the open-system device file for the same logical volume. Table 3-10 describes the FX volume association parameters. Figure 3-6 through Figure 3-13 show the structure and contents of the FX volume definition file for each supported platform.

Note: Regarding the specification of the same VSN in volume definition file for 01-XX-48 or later: The same VSN can be defined in the Volume definition file using VSN identification, and both volumes can be used by FXotm and FXmto. The definitions should be defined with 35 digits, using the alphabet (A-Z, @, #, and \) or numeral (0-9) characters.

The -A, -B, and -C FX volumes and the OPEN-*x*-FXoto volumes can be defined in the same FX volume definition file. ***For example:***

```
XXX/XXXXXX MVS01 3390-3A
YYY/YYYYYY VSN01 OPEN-3
end
```

Table 3-10 FX Volume Association Parameters

Number	Name	Function	Enter
1	Device file name	Specifies raw device (partition) name defined for open-system.	Character-type device file name (e.g., **c1t0d2** for HP-UX, **c1t0d2s1** for Solaris.
2	VOLSER	Specifies logical volume defined for z/OS.	Six-character volser (e.g., FX45). A volser can use the following characters: A-Z, 0-9, @, #, \
3	LVI or LU type (emulation)	Specifies LVI or LU type of FX volume.	Correct LVI/LU for FX volume: 3390-3A, -3B, -3C, 3380-KA, -KB, -KC or OPEN-*x*-FXoto. Make sure to define all OPEN-*x* FMT volumes in a separate file.
4	Carriage return	Marks end of parameter set.	Make sure to press the **Return** key (**Enter** key for Windows 2000/2003/Windows NT) at the end of each line.
5	End of file	Marks end of parameter file.	**end**

To create the FX volume definition file:

1. Open a new empty text file. For UNIX-based systems, use the UNIX vi editor (e.g., **vi datasetmount.dat**). For Windows 2000/2003/Windows NT systems, use any text editor, and make sure to use plain text. The file name must be **datasetmount.dat** (all lowercase), and the file must be located in the current working directory when you start FCU. If you are creating two FX volume definition files, use **datasetmount1.dat** and **datasetmount2.dat**, and remove the “1” or “2” from the desired file before starting FCU.

2. Add the volume association parameters for the FX volumes to the file.
   – Put at least one space between each parameter, and press the **Return** key at the end of each line to separate the parameter sets. All three parameters (device name, volser, LVI type) are case-sensitive. If you add comments to the file, make sure that each comment line starts with **#**. Make sure to enter **end** on the last line of the file.
3. When you are done adding the volume association parameters for each FX volume to the volume definition file, save your changes and exit the text editor.

```
/dev/rdsk/cx1tx2dx3sx4    AAAAAA  3390-3A   MFN  MVS1
/dev/rdsk/cy1ty2dy3sy4    AAAAAA  3390-3A   MFN  VOS3
/dev/rdsk/cz1tz2dz3sz4    cccccc  3380-KB
/dev/rdsk/cw1tw2dw3sw4    dddddd  3380-KA
       (1)                  (2)     (3)     (4)   (5)    (6)
end
(7)
```

Figure 3-6 FX Volume Definition File for Solaris (mto/otm Shown)

Note: x = controller number, y = SCSI target ID (TID), z = LUN, w = partition (or slice)

```
/dev/rdsk/cx1tx2dx3       AAAAAA  3390-3A   MFN  MVS
/dev/rdsk/cy1ty2dy3       AAAAAA  3390-3A   MFN  VOS3
/dev/rdsk/cz1tz2dz3               cccccc  3380-KB
/dev/rdsk/cw1tw2dw3               dddddd  3380-KA
       (1)               (2)       (3)     (4)   (5)    (6)
end
(7)
```

Figure 3-7 FX Volume Definition File for HP-UX (oto Shown)

Note: In cxtydz, x = controller number, y = SCSI TID, z = LUN. In OPEN-x, x = 3, 8, K,E, L, M, 9 or V.

```
/dev/rhdiskn1             AAAAAA          3390-3A   MFN  MVS
/dev/rhdiskn2             AAAAAA          3390-3A   MFN  VOS3
/dev/rhdiskn3             cccccc          3380-KB
/dev/rhdiskn4             dddddd          3380-KA
  (1)              (2)              (3)     (4)  (5)   (6)
end
(7)
```

Figure 3-8 FX Volume Definition File for IBM AIX (mto/otm Shown)

Note: n = disk ID number (note that the first, second, and third drives are 0, 1, 2).

```
/dev/rrzX1Y1Z1    AAAAAA  3390-3A   MFN   MVS
/dev/rrzX3Y3Z3    cccccc  3380-KB   MFN   VOS3
/dev/rrzX4Y4Z4    dddddd  3380-KA
      (1)           (2)     (3)     (4)    (5)    (6)
end
(5)
```

Figure 3-9 FX Volume Definition File for DIGITAL Tru64 UNIX (oto Shown)

Note: X = b through h = LUN1 through LUN7 (no letter is used for LUN0); Y = fibre bus number × 8 + SCSI TID; Z = partition = a through h. For example, rrzc18a = SCSI TID 2, LUN2 (partition a) on fibre bus 2.

Note: OPEN-x = 3, 8, K, E, L, M , 9 or V

```
\\.\PHYSICALDRIVE0        AAAAAA  3390-3A   MFN MVS
\\.\PHYSICALDRIVE1        AAAAAA  3390-3A   MFN VOS3
\\.\PHYSICALDRIVE2 cccccc  3380-KB
\\.\PHYSICALDRIVE3        dddddd  3380-KA
         (1)                (2)     (3)     (4)  (5)   (6)
end
(7)
```

Figure 3-10 FX Volume Definition File for Windows NT /2000/2003 Systems (mto/otm Shown)

Note: n = disk ID number.

```
/dev/rdsk/cx1tx2dx3s0     AAAAAA    3390-3B  MFN  MVS
/dev/rdsk/cy1ty2dy3s0     AAAAAA    3390-3A  MFN  VOS3
/dev/rdsk/cz1tz2dz3s0     cccccc    3380-KB
/dev/rdsk/cw1tw2dw3s0     dddddd    3380-KA
          (1)                       (2)       (3)     (4)  (5)  (6)
end
(7)
```

Figure 3-11 FX Volume Definition File for NCR® UNIX

```
/dev/rdsk/sdx1  AAAAAA  3390-3A  MFN  MVS
/dev/rdsk/sdx2  AAAAAA  3390-3A  MFN  VOS3
/dev/rdsk/sdx3  cccccc  3380-KB
/dev/rdsk/sdx4  dddddd  3380-KA
      (1)       (2)    (3)     (4)  (5)   (6)
end
(7)
```

Figure 3-12 FX Volume Definition File for DYNIX/ptx®

```
/dev/rsda     AAAAAA       3390-3A    MFN   MVS1
/dev/asdb     AAAAAA       3390-3A    MFN   VOS3
/dev/rsd      cccccccc     3380-KB
/dev/rsd      ddddddd      3380-KA
    (1)           (2)         (3)     (4)   (5)        (6)
end
```

(7)

Figure 3-13 FX Volume Definition File for Linux

Verifying Mainframe Dataset Requirements

FAL and FCU have specific requirements for the FX source and target datasets. Table 3-11 specifies the requirements for FX datasets. The FCU GUI (see sections Performing File Transfer Operations - UNIX and Performing File Transfer Operations – Windows) allows the user to display the dataset attributes and verify the dataset requirements. FCU for UNIX (version 01-01-41 and later) also provides the **listvol** function to display z/OS dataset attributes without using the GUI. The FXotm target dataset (which can also be an FXoto intermediate dataset) must be created and properly configured before the FX operation is performed. FCU does not support automatic expansion of the extent during FXotm operations. The FX ALC utility allocates intermediate datasets in accordance with the requirements specified below.

Table 3-11 z/OS Dataset Requirements

Item	Requirement(s)
Dataset organization (DO) type	SAM (sequential-access method). FAL/FCU does not support any other DO types (e.g., DAM, VSAM, PAM). If a non-SAM dataset is specified, FAL/FCU will return an error. Multiple-volume datasets are not supported. FAL/FCU can only process the portion within one logical volume.
Dataset name	No spaces. If FAL/FCU encounters a space, it will accept the characters before the space as the dataset name and continue processing.
Record format (RF)	Fixed-length or variable-length record format. FAL/FCU does not support undefined-length or spanned record formats. If an illegal RF is detected, FAL/FCU will return an error. No key. If a record with a key is accessed, FAL/FCU will return an error. For FXotm, the record format of the target dataset must be preconfigured to match the record format of the data entities in the source file. For VSE source and target datasets, the VSE record option must be used to specify the RF.
Block length (BL)	Any length within the extent supported by the OS. If an illegal BL is detected, FAL/FCU will return an error. For FXotm, the block length of the target dataset must be preconfigured to match the block length of the data entities in the source file. For VSE source and target datasets, the VSE record option must be used to specify the BL.
Record length (RL)	Any length within the extent supported by the OS. If an illegal RL is detected, FAL/FCU will return an error. ***Note:*** FAL/FCU cannot process a variable-length dataset which includes a record with no data entity (RL = 4). For FXotm, the record length of the target dataset must be preconfigured to match the record length of the data entities in the source file. For VSE source and target datasets, the FCU VSE record option must be used to specify the RL.
Track format	Standard record 0 (R0). FAL/FCU cannot process tracks with nonstandard R0.
VTOC	For MVS: standard or index VTOC. For an index VTOC, FAL/FCU ignores the index and accesses the entire VTOC sequentially. For VSE: The user must specify the RF, BL, and RL using the FCU VSE record option. ***Note:*** The FAL functions cannot be used on VSE datasets.
Database file	Direct access is not supported; must be converted to a SAM file.

Allocating FXoto Intermediate Datasets

When you perform FXoto operations using OPEN-*x* FMT volumes, you must allocate the intermediate datasets before starting the file transfer operations. The FX Allocater (ALC) utility can only be used on OPEN-*x* volumes which have already been formatted using the FX FMT utility.

Note: FCU version 01-01-41 or later is required for the ALC utility.

Note: For versions 01-01-41: The ALC utility for UNIX is a UNIX command executed from the UNIX command line. The ALC utility for Windows 2000/2003/Windows NT systems is a GUI. The ALC utility for UNIX can only be used on volumes formatted with the FMT utility for UNIX. The ALC utility for Windows 2000/2003/Windows NT systems can only be used on volumes formatted with the FMT utility for Windows 2000/2003/Windows NT systems.

CAUTION: The capacity of the intermediate dataset varies depending on block length so remember to calculate the required size for the intermediate dataset. When you transfer variable-length records, make sure to take the four-byte RL information and four-byte BL information for each record into account.

UNIX - To allocate an FXoto intermediate dataset using the ALC utility:

1. Log in to the system as **root**.
2. Enter the following command at the UNIX command line prompt:
   # allocds -d *devname* [-n *datasetname*] [-f *recform*] [-r *reclen*] [-b *blocklen*] [-c *cylinders*]

   Note: Enter only one value for each parameter. You can only allocate one dataset at a time.

   -d devname: Specify the raw device name of the OPEN-*x* volume on which the dataset is being allocated. This parameter is required and must be specified.

   -n datasetname: Specify the name of the dataset being allocated (maximum forty-four characters: A-Z, 0-9, @, #, ., \). Use uppercase letters only, and do not use any spaces or symbols other than @, #, ., and \. This parameter is required. If not specified, ALC will return the residual capacity (free space) on the specified volume in number of cylinders.

   -r recform: Specify the record format of the dataset being allocated: **F** (fixed-length and de-blocking), **FB** (fixed and blocking), **V** (variable and de-blocking), or **VB** (variable and blocking). This parameter is required. If not specified, the default value of **F** is used.

   -r reclen: Specify the record length (decimal) of the dataset being allocated: **1 to 32760**. This parameter is required. If not specified, the default value of 4096 is used.

-b blocklen: Specify the block length (decimal) of the dataset being allocated.

When record format = F, block length = record length.

When record format = FB, block length = record length × N (N = integer).

When record format = V/VB, block length = record length + 4 or more.

This parameter is required. If not specified, the following default values are used:

When record format = F/FB, default block length = record length.

When record format = V/VB, default block length = record length + 4.

-c cylinders: Specify the size of the dataset being allocated in number of cylinders (decimal). This parameter is required. If not specified, the default value of 100 is used.

3. If the ALC allocate operation could not be started due to an error condition, the **Allocate check error** message is displayed. If the ALC allocate operation did not complete successfully, an error message is displayed. Remove the error condition, and retry the operation. See section C.2 in Appendix C for further information on errors in UNIX.

Windows 2000/2003/Windows NT systems**:** To allocate an intermediate FXoto dataset using the ALC utility:

1. Log in to the system as administrator.
2. Double-click on the **Allocate** icon to start the ALC utility and open the Allocation panel.
3. The ALC utility automatically displays the first OPEN-*x* FMT volume (in alphanumeric order) in the **VOLSER** field. If this is not the desired volume, select the desired volume from the drop-down list of volsers. If ALC could not find any OPEN-*x* FMT volumes, ALC displays the **FX format disk not found** message.
4. Enter the name of the dataset being allocated in the **Dataset** field (maximum forty-four characters: A-Z, 0-9, @, #, ., \). Do not use any spaces or symbols other than @, #, ., and \.
5. Enter or select the size of the new dataset (number of cylinders, number of tracks) in the **Cylinder** and **Track** fields. The file size will be (# of cyl) + (# of tracks). The **Max.** button enters the maximum size for the new dataset in the **Cylinder** and **Track** fields based on the available capacity. The **Available Capacity** box displays the free space on the specified volume, so that you can select the appropriate size for the new dataset.
6. Enter or select the record format in the **Record format** field: F, FB, V, or VB.
7. Enter or select the record length in the **Record length** field:
   - When record format = **F**, record length = block length.
   - When record format = **FB**, record length = block length ÷ N (N = integer).

– When record format = **V or VB**, 5 ≤ record length ≤ (block length – 4).

8. Enter or select the block length in the **Block length** field. If block length = record length, select the **Copy** button to copy the record length into the **Block length** field.
   – When record format = **F or FB**, 1 ≤ block length ≤ 32760.
   – When record format = **V or VB**, 9 ≤ block length ≤ 32760.
9. When all parameters for the new dataset are correct, select the **Start** button.
10. When the allocate operation completes successfully, the **Allocation complete** message is displayed. If the allocate operation could not be started due to an error condition, the **Allocate check error** message is displayed. If the allocate operation did not complete successfully, one of the following error messages is displayed:

Open error! (n)	Open process error on the FXoto volume.
Seek error! (n)	Seek process error on the FXoto volume.
Read error! (n)	Read process error on the FXoto volume.
Write error! (n)	Write process error on the FXoto volume.
Close error! (n)	Close process error on the FXoto volume.

11. When you are finished allocating datasets on FXoto volumes, select **Close** to close the Allocation panel and exit the ALC utility.

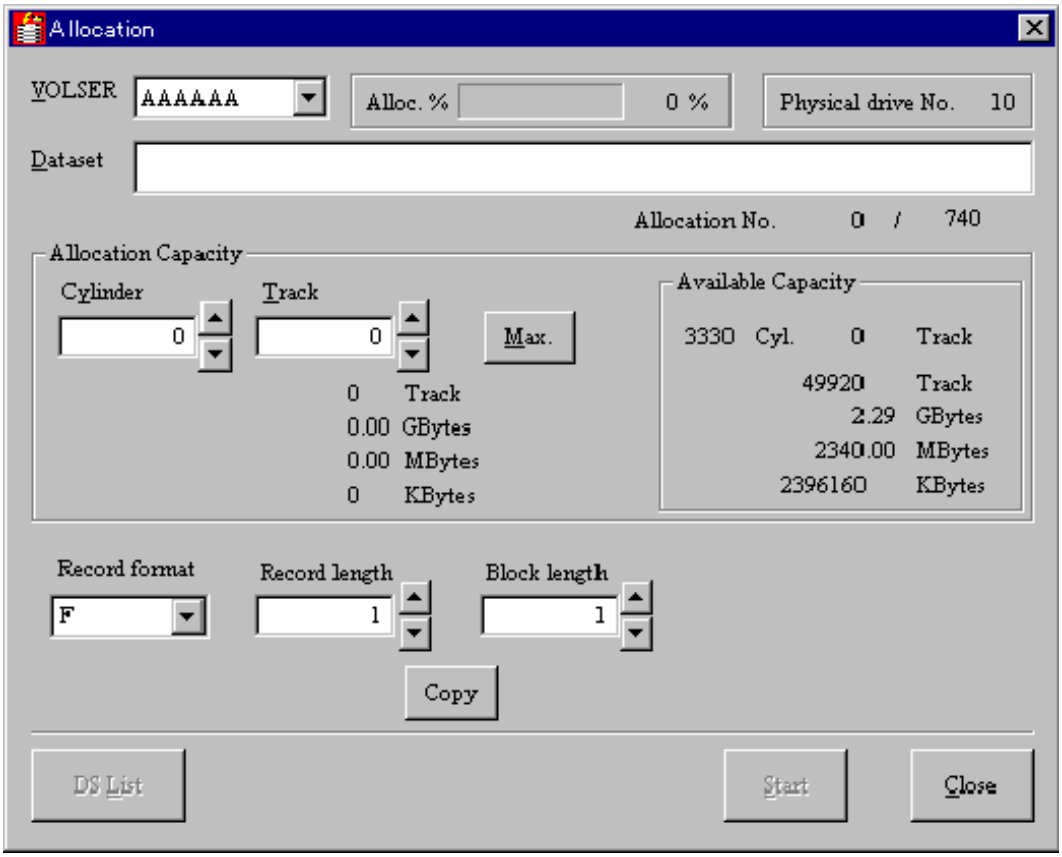


Figure 3-14 ALC Utility for Windows 2000/2003/Windows NT systems

Figure 3-15 ALC Disk Not Found Message

Figure 3-16 ALC Allocation Complete Message

Figure 3-17 ALC Error Message

4

Using the Cross-OS File Exchange Software

This chapter describes the user interface and the commands available to an FX user.

- FCU for UNIX
- FCU for Windows
- Format Utility for Windows
- Allocation Utility for Windows

FCU for UNIX

The FCU GUI enables you to perform FX file transfer operations interactively and provides access to detailed information on the datasets/files in the specified FX source volume/directory. The FCU GUI displays the FX operations in the FCU parameter definition file (if specified), allows you to modify the FCU parameter definition file interactively, and also allows you to enter FCU parameters and perform FX operations manually. The FCU GUI also displays the error information for FX operations.

FCU Version and Copyright Screen

To start the FCU GUI program for UNIX -based platforms:

1. At the UNIX command line prompt, enter: **fcu [-nc] [param]**

   The **-nc** option (**nc** = no checking) tells FCU to execute all specified FX operations without requesting confirmation for FCU parameters or checking for existing FXmto target files. If you want to bypass these confirmations, enter **-nc**.

   The **param** option tells FCU whether to use the FCU parameter definition file or a specific FCU initiation parameter set to perform FX operations. The **param** option must have one of the following three values:

   – [blank]. If you want to use the default FCU parameter definition file (**fcudata.param** in the current directory), leave the **param** option blank (do not enter anything).

   – **file_name**. If you want to use a different FCU parameter definition file, enter the file name with complete path (absolute or relative) if not in the current directory.

   – **-P + parameters**. If you want to perform one specific FX operation, enter **-P** followed by the FCU initiation parameter set (e.g., **mto VSN:dataset targetfile No No No**) for the desired FX operation. The **-P** option requires the **-nc** option.

   For example:

   – If you want to use the default FCU parameter definition file and check the parameters and FXmto target files, enter: **fcu**

   – If you want to use the default FCU parameter definition file and perform all operations without checking parameters or FXmto target files, enter: **fcu -nc**

   – If you want to use a different FCU parameter definition file and perform all operations without checking parameters or FXmto target files, enter: **fcu -nc filename**

   – If you want to perform one specific FX operation, enter: **fcu -nc -P [parameters]**

Note: The following warnings may appear during FCU startup. These warnings do not affect FCU and can be ignored.

WARNING: Missing characters in String to FontSet conversion.

WARNING: Cannot convert string “-dt-interface system-medium-r-normal -m*-*-*-*-*-*-*-*-*” to type FontSet.

2. The FCU GUI program now starts loading. The FCU version and copyright screen (see Figure 5-1) is displayed while FCU is loading. When FCU is finished loading, the FCU main panel is displayed.
3. If you specified the **-nc** option, FCU processes all specified operations, overwrites existing mto target files, terminates, and displays any error information at the UNIX prompt.

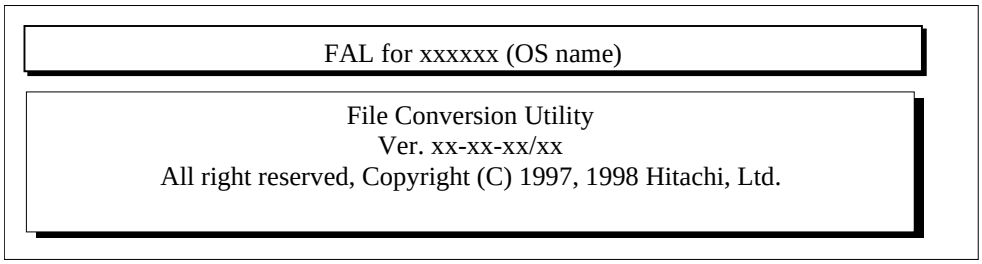


Figure 4-1 FCU Version and Copyright Screen (UNIX)

File Conversion Utility Screen

The FCU main panel opens when the FCU GUI program is finished loading. The FCU main panel displays the FCU initiation parameter sets in the specified FCU parameter definition file (if available), allows you to perform FX operations, and provides access to all FCU functions.

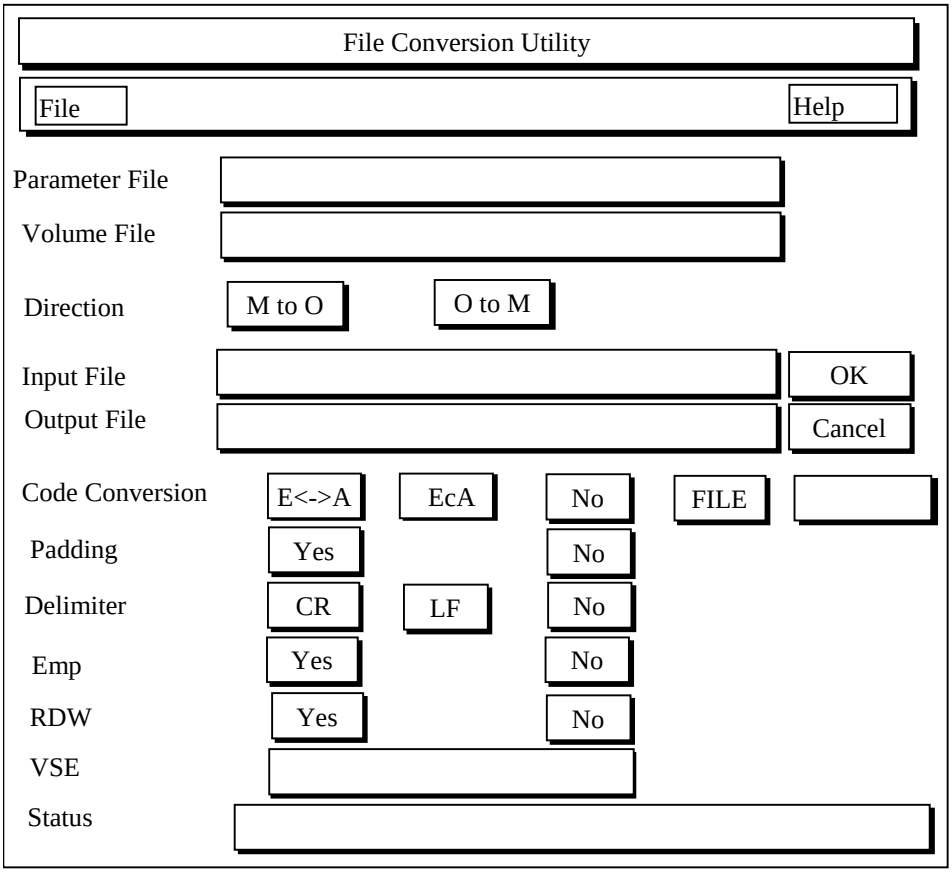


Figure 4-2 FCU Main Panel for UNIX Platforms

The **File** and **Help** buttons display the File menu commands and Help menu commands. These commands are described later in this section.

The **Parameter File** field displays the FCU parameter definition file that you specified by the **param** option when you started FCU. If this field is blank, FCU could not find the default or specified FCU parameter definition file. If you want to use an FCU parameter definition file, you can enter the desired file name in this field (complete path if not in the current directory). If you do not want to use an FCU parameter definition file, you can leave the **Parameter File** field blank and enter the FCU initiation parameters manually.

The **Volume File** field displays the FX volume definition file. This file must have the default name and location (**datasetmount.dat** in current directory). If this field is blank, FCU could not find the file and will not be able to perform FX operations. In this case, exit FCU, and create the FX volume definition file as described in section “Creating the FX Volume Definition File(s)”.

When FCU starts up, the first set of FCU initiation parameters is automatically loaded from the specified FCU parameter definition file (unless the file is not found). If desired, you can change any of the parameters, or you can use the **File-Load** command to load the next parameter set. The FCU initiation parameters are:

- **Direction**. The **Direction** buttons allow you to select the desired direction for the FX operation: **M to O** = FXmto, **O to M** = FXotm.
- **Input File**. The **Input File** field allows you to enter the name of the FX source file. For FXmto, enter the z/OS volser and dataset name (**VSN:dataset**). For FXotm, enter the UNIX file name (with complete path if not in the current directory).
- **Output File**. The **Output File** field allows you to enter the name of the FX target file. For FXmto, enter the UNIX file name (with complete path if not in the current directory). For FXotm, enter the VSN and dataset name (**volser:dataset**).
- **Code Conversion**. The **Code Conversion** buttons allow you to select the desired code conversion option (see section Code Conversion (CC) Option): **E<->A** = default code conversion table, **EcA** = default code conversion table (for FXoto only), **No** = no code conversion, **File** = enter the file name of your conversion table (with complete path if not in current directory).
- **Padding**. The **Padding** buttons allow you to select the desired padding option (see section Padding (PAD) Option): **Yes** = padding, **No** = no padding.
- **Delimiter**. The **Delimiter** buttons allow you to select the desired delimiter option (see section Delimiter (DEL) Option): **CR** = carriage return, **LF =** line feed, **No =** no delimiters.
- **Emp**. The **Emp** buttons allow you to select the empty file option (see section Empty File (Emp) Option):
  Yes = source file is empty, **No** = source file is not empty.
- **RDW**. The **RDW** buttons allow you to select the record description word option (mto only) (see section Empty File (Emp) Option): **Yes** = add RDW to each record (**Code Conversion**, **Padding**, and **Delimiter** must be **No**), **No** = do not add RDW to each record.
- **VSE.** The VSE field allows you to enter the VSE record information: **RF,RL,BL** Use a comma (no spaces) between each value. See section Record Description Word (RDW) Option for further information on the VSE record option values.

The **OK** button starts the specified FX operation. The **Cancel** button removes the values entered by the user and returns the FCU main panel to the initial settings. (The **Cancel** button does not cancel the FX operation in progress.) Be careful not to click **OK** or **Cancel** more than once. The **Status** field displays the status of the requested FX operation:

- **Now checking** = FCU is executing a dataset search or file attribute check. If you specified the **-nc** option when you started FCU, this check does not occur.
- **Overwrite ? (OK/Cancel)** is displayed if the FXmto target file already exists. Click **OK** to overwrite the existing file, or click **Cancel** to cancel the requested operation. If you specified the **-nc** option when you started FCU, this confirmation does not occur.
- **x%** = The requested FX operation is *x*% complete.
- **Complete** = The requested FX operation completed successfully.
- **Error**. The **Status** field also displays error information for FCU and FX operations. See Appendix C for further information on error conditions.

File Menu Commands

The **File** menu provides access to the following FCU functions:

- **Load**. This command loads the parameter sets from the specified FCU parameter definition file onto the FCU main panel. Each time you select **Load**, the next set of parameters is loaded. If you enter a file name in the **Parameter File** field, the **Load** command opens the file and loads the first parameter set (or creates the new file). If the FCU parameter definition file is empty or was not found, FCU ignores this command.
- **Save**. This command saves the FCU parameter definition file. If no FCU parameter set was previously loaded, the current parameter set is added to the file. If a parameter set was previously loaded and you made changes, the current parameter set overwrites and replaces the previously loaded parameter set. If you make changes and do not select **Save**, FCU will discard your changes when you select **Load** or **Exit**.
- **Delete**. This command deletes the currently loaded parameter set from the FCU parameter definition file. If the FCU parameter definition file does not yet exist or does not contain the parameter set on screen, FCU ignores this command.
- **Exit**. This command closes the current FCU parameter definition file (unsaved changes are discarded), and then closes the FCU program.

Help Menu Commands

The **Help** menu provides access to the following FCU functions.

Note: When installing FX Code Converter, the Edit_prm menu is displayed, and the parameters for FX Code Converter can be displayed.

- **Volume**. This command displays the contents of the FX volume definition file, so that you can verify that the FX volumes are properly defined.
- **MF-File**. This command displays the dataset information for each dataset in the specified mainframe (MF) volume. The VSN must be entered in the **Input File** field (for FXmto) or **Output File** field (for FXotm) on the FCU main panel.
  - Dataset name: An asterisk (*) before the dataset name indicates that FAL/FCU can process the dataset. A dash (-) indicates that FAL/FCU cannot process the dataset. A question mark (?) indicates that FCU can process the dataset only if the VSE record option is used to specify the RF, RL, and BL.
  - Dataset organization (DO) type: SAM, DAM, PAM, VSAM, ??? = unknown.
  - Record format (RF): F = fixed length, V = variable length, U = undefined length, S = spanned record, ? = unknown.
  - Block length (BL): in bytes
  - Record length (RL): in bytes

- Dataset size (DS): in tracks

- **UX-File**. This command displays the UNIX (UX) files in the directory specified in the **Input** or **Output File** field on the FCU main panel. If no directory is specified in the **Input File** or **Output File** field, FCU displays the files in the current directory. If a nonexistent directory is specified, FCU will return an error.
- **Error**. This command opens the error information panel, which displays the FAL, FCU, and system error codes/messages.
- **OnVersion**. This command displays the FCU version and copyright information screen.

```
/dev/rdsk/cxtydz  volser  3390-3B
/dev/rdsk/cxtydz  volser  3390-3A
/dev/rdsk/cxtydz  volser  3390-3C:
/dev/rdsk/cxtydz  volser  3380-KB
/dev/rdsk/cxtydz  volser  3380-KA
/dev/rdsk/cxtydz  volser  3380-KC
 :
end
```

Figure 4-3 Help-Volume Display (HP-UX Shown)

```
Dataset Information : VSN = xxxxxx   Device Emulation Type = 3390-3B

 Dataset Name           DO      RF    BL      RL      DS

*SAMFILE01.FIX          SAM     F     4096    128     150   ← Can be processed by FCU.
-DAMFILE.F              DAM     F     4096    128     30    ← Cannot be processed by FCU.
*SAMFILE02.VAR          SAM     V     4000    80      50    ← Can be processed by FCU.
-PAMFILE                PAM     F     5000    100     200   ← Cannot be processed by FCU.
-VIRTUALSTORAGEACCESS   VSAM    V     32768   4096    3000  ← Cannot be processed by FCU.
-UNDEFSAMFILE           SAM     U     8000    200     80    ← Cannot be processed by FCU.
-SAMFILESPANNED         SAM     S     8192    8192    300   ← Cannot be processed by FCU.
```

Figure 4-4 Help MF-File Display

```
            UNIX FILE LIST : DIR = /aaaaa/bbbbb/cccc

ddddd.dd       eeeeee       ffffff.fffffff
hhhhh.hhhh     zzzzzz.z     xxxx.x
YYYYYYYYY
```

Figure 4-5 Help UX-File Display

Error Information Screen

This window opens when the Error command is used. It displays the FAL, FCU, and system error codes/messages.

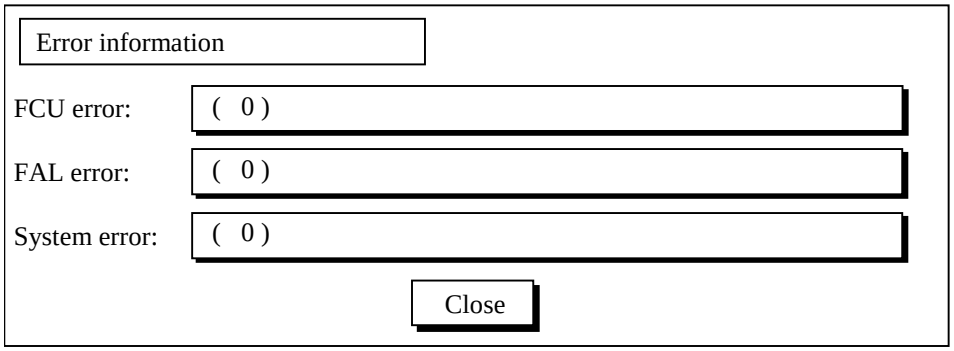


Figure 4-6 Error Information Display

FCU for Windows

FCU Version and Copyright Dialog

To start the FCU GUI program for Windows 2000/2003/Windows NT systems:

1. Log on with Administrator access privileges.
2. Start the FCU GUI as follows: Click **Start-Programs-FCU-FCU**, or open the **c:** folder and double-click on **FCU**, or create a shortcut for **FCU** on the desktop.

   Note: Do not start FCU by dragging and dropping an FCU parameter definition file on the FCU program icon. FCU program operation cannot be guaranteed.
3. If you want to specify any of the FCU options, start FCU from the command line (DOS prompt) as follows: Go to the FCU directory (containing **fcu.exe** and **datasetmount.dat**), and enter **fcu [-nc] [-cl] [param]**

   The **-nc** option is the same as for UNIX: All specified FX operations are performed without confirmation of FCU parameters or FXmto target file overwrites.

   The **-cl** option specifies that all FCU log files will be cleared before starting.

   The **param** option is the same as for UNIX:

   – If you want to open a new untitled FCU parameter definition file when you start FCU, leave the **param** option blank.

   – If you want to load an FCU parameter definition file when you start FCU, enter the file name with complete path if the file is not in the current directory.

4. The FCU GUI program now starts loading. The FCU version and copyright screen (see Figure 5-6) is displayed while FCU is loading. When FCU is finished loading, the FCU main panel is displayed (see section Performing File Transfer Operations (UNIX)).
5. If you started FCU from the DOS prompt and specified the **-nc** option, FCU processes all specified operations, overwrites existing FXmto target files, and then terminates and displays any error information at the DOS prompt.

Figure 4-7 FCU Version and Copyright Screen (Windows 2000/2003/Windows NT Systems)

File Conversion Utility Window

The FCU main panel opens when FCU is finished loading. The FCU main panel displays the FCU parameter definition file (or **Untitled** if no file was specified), allows you to perform FX operations, and provides access to all FCU functions.

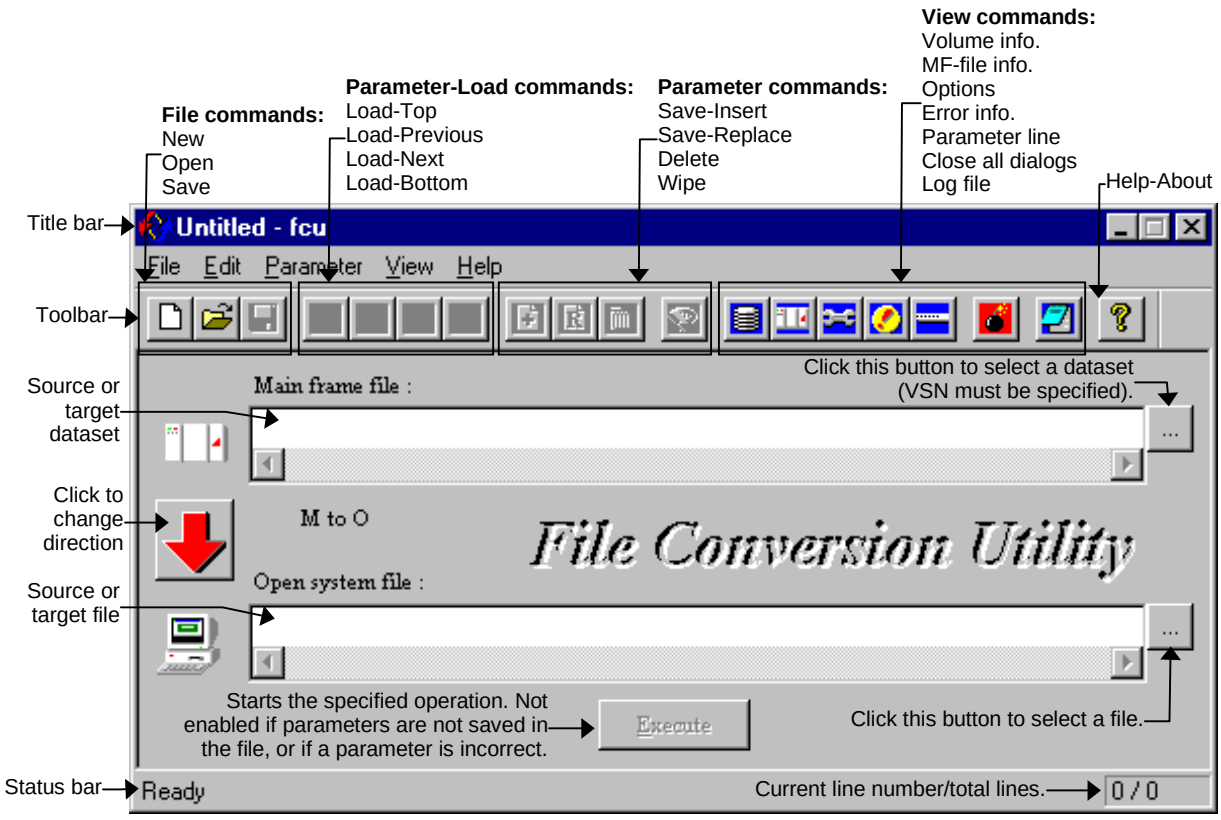


Figure 4-8 FCU Main Panel for Windows 2000/2003/Windows NT Systems

The FCU title bar displays the current FCU parameter definition file. The toolbar provides speed buttons for the commonly used FCU functions. The status bar displays the current line number and total number of lines in the current FCU parameter definition file. The **Main frame file** and **Open-system file** fields display the files to be transferred (no spaces allowed). The file selection buttons ( ) allow you to select the desired MF dataset and NT file.

The **File** menu provides access to the following FCU functions:

- The **File-New** command (Ctrl+N) opens a new FCU parameter def. file (untitled.prm).
- The **File-Open** command (Ctrl+O) opens an existing FCU param. def. file (filename.prm).

- The **File-Save** command (Ctrl+S) saves the current FCU parameter definition file. Deleted and replaced lines are discarded, inserted lines are added, and all lines after **end** are discarded.

  Note: This command does not save the current parameter set.

- The **File-Save As...** command saves the current FCU parameter definition file with a different file name and/or location.
- The **File-Exit** command (Ctrl+X) exits the FCU software.

The **Edit** menu is reserved for future enhancement and is not yet enabled.

The **Parameter** menu provides access to the following FCU functions:

- The **Parameter-Load** command loads the **Previous**, **Next**, **Top**, and **Bottom** parameter lines from the current FCU parameter definition file. The FCU main panel status bar updates the current line number when any **Parameter-Load** command is executed.
- The **Parameter-Save** command allows you to either **Insert** the parameter set being displayed into the current FCU parameter definition file, or **Replace** the current parameter set (previously loaded) with the parameter set being displayed. If you do not select this command, your parameter changes will not be saved.

  Note: This command does not save the current FCU parameter definition file (you must use **File-Save/Save As**).

- The **Parameter-Delete** command deletes the current parameter set from the current FCU parameter definition file. The line is not permanently deleted until you save the current FCU parameter definition file using the **File-Save** command.
- The **Parameter-Wipe** command clears all FCU initiation parameters displayed on screen, so that you can input new parameters easily. This command does not delete the current parameter set.

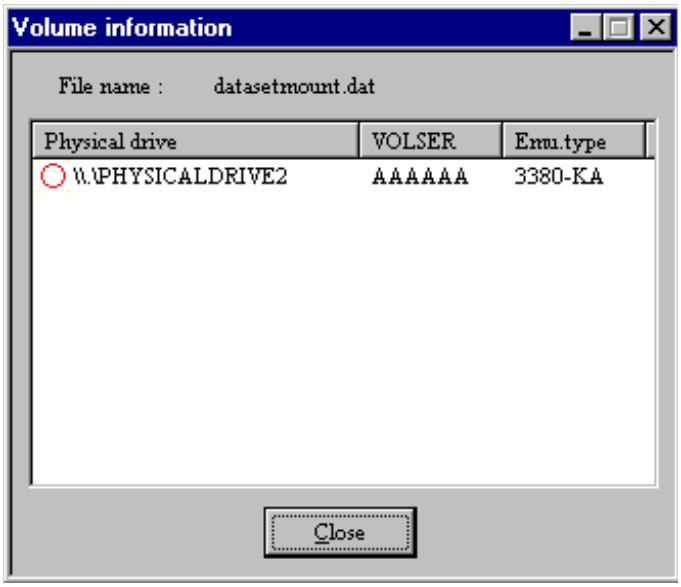
The **View** menu provides access to the following FCU functions:

- The **View-Toolbar** and **View-Status bar** commands display/hide the FCU toolbar and status bar.
- The **View-Volume information...** command opens the FCU Volume Information panel, which displays the contents of the FX volume definition file.
- The **View-MF-file information...** command displays the following information for the mainframe (MF) files (datasets) in the volume specified in the **Mainframe file** field on the FCU main panel.
- The **View-Option...** command opens the Option panel, which allows you to view/change the FCU file transfer options.
- The **View-Error information...** command opens the Error Information panel.

- The **View-Parameter line...** command opens the Parameter Line panel.
- The **View-Close all dialogs** command closes all open panels except the FCU main panel.
- The **View-Log file** command opens the log file for the current FCU parameter definition file.
- The **Help-About FCU...** menu command opens the FCU version and copyright screen (refer to Figure 5-6).

Volume Information Dialog

The **View-Volume information...** command opens the FCU Volume Information panel, which displays the contents of the FX volume definition file. A ○ displayed next to a volume indicates that the volume definition is correct and FCU can access the volume. An **8** displayed next to a volume indicates that the volume definition is not correct and FCU cannot access the volume.



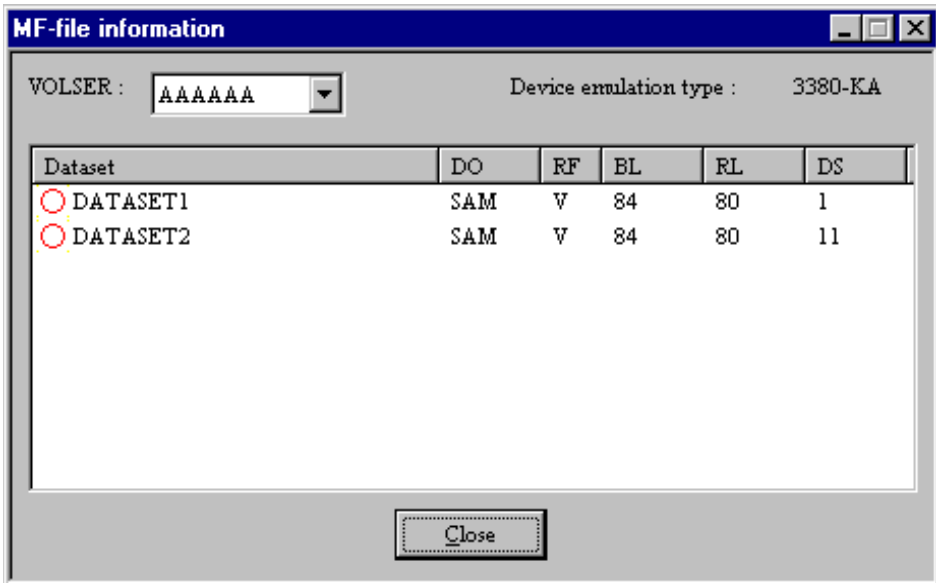
Note: This example shows an FX volume definition file which defines only one FX volume.

Figure 4-9 Volume Information Panel

Mainframe File Information Dialog

The **View-MF-file information...** command displays the following information for the mainframe (MF) files (datasets) in the volume specified in the **Mainframe file** field on the FCU main panel:

- ○ = the dataset can be processed by FCU.
  8 = the dataset cannot be processed by FCU.
  ? = the dataset can be processed by FCU only if the VSE record option is used to specify the RF, RL, and BL.
- **Dataset** = dataset name
- **DO** = dataset organization type: **SAM**, **DAM**, **PAM**, **VSAM**, **?** (other than above)
- **RF** = record format: **F** (fixed-length), **V** (variable-length), **U** (undefined), **S** (spanned), **?** (other than above)
- **BL** = block length
- **RL** = record length
- **DS** = dataset size (in number of tracks)



Note: This example shows an FX volume (3380-KA, VSN = AAAAAA) which contains only two datasets.

Note: When this panel is opened using the **Mainframe file** selection button ( ), only SAM datasets are displayed.

Figure 4-10 MF-File Information Panel

Option Dialog

The **View-Option...** command opens the Option panel, which allows you to view/change the FCU file transfer options (code conversion, padding, delimiter, empty file, RDW, and VSE), continuous execution option, and clear log file option.

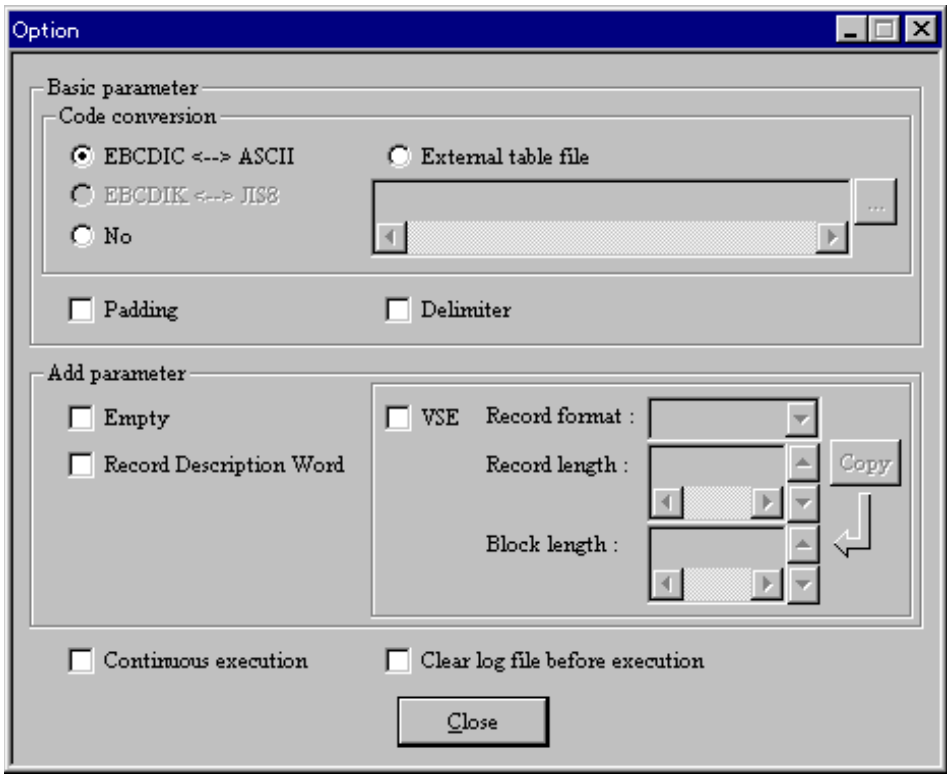


Figure 4-11 Option Panel

- Select the **EBCDIC<-->ASCII** button to use the default code conversion table (**EA**, **EcA**). Select **No** for no code conversion (**No**). Select **External table file** and enter the file name with path if not in the current directory (e.g., **/directory/filename.tbl**).
- Check the **Padding** box to request the padding option (**Yes**).
- Check the **Delimiter** box to request the delimiter option (**Yes**).
- Check the **Empty** box to request the empty file option (**Emp=Yes**).
- Check the **Record Description Word** box to request the RDW option (**RDW=Yes**). ***Note:*** If the **Record description word** box is checked, the **EBCDIC<-->ASCII** button and the **Padding** and **Delimiter** boxes are ignored.
- Check the VSE box and enter the RF, RL, and BL to request the VSE record option.

– Check the **Continuous execution** box to tell FCU to process the rest of the FCU initiation parameter sets in the specified FCU parameter definition file without stopping after each completed operation (equivalent to the **-nc** option starting at the desired line). FCU will execute all lines from the current line to the **end**. If you do not check the **Continuous execution** box, FCU will stop after each operation.

– Check the **Clear log file before execution** box to clear the log file for the current FCU parameter definition file (e.g., fcudata.prm.log). The user should clear the FCU log files as needed to decrease the file size and save space on the current drive.

Parameter Line Dialog

The **View-Parameter line...** command opens the Parameter Line panel, which displays the current line (parameter set) in the current FCU parameter definition file.

The **Parameter** menu provides access to the following FCU functions:

- The **Parameter-Load** command loads the **Previous**, **Next**, **Top**, and **Bottom** parameter lines from the current FCU parameter definition file. The FCU main panel status bar updates the current line number when any **Parameter-Load** command is executed.
- The **Parameter-Save** command allows you to either **Insert** the parameter set being displayed into the current FCU parameter definition file, or **Replace** the current parameter set (previously loaded) with the parameter set being displayed. If you do not select this command, your parameter changes will not be saved.

  Note: This command does not save the current FCU parameter definition file (you must use **File-Save / Save As**).
- The **Parameter-Delete** command deletes the current parameter set from the current FCU parameter definition file. The line is not permanently deleted until you save the current FCU parameter definition file using the **File-Save** command.
- The **Parameter-Wipe** command clears all FCU initiation parameters displayed on screen, so that you can input new parameters easily. This command does not delete the current parameter set.

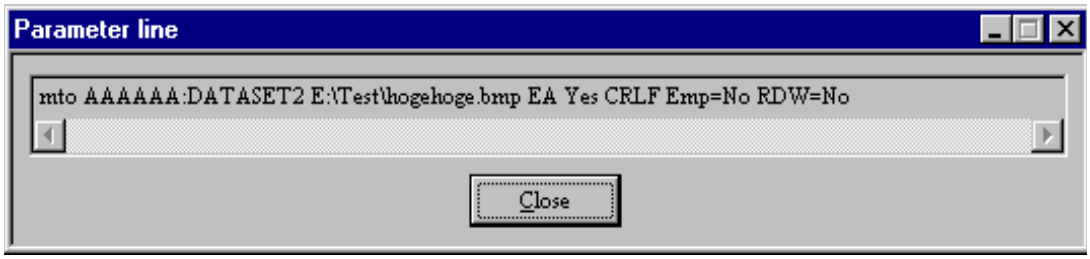


Figure 4-12 Parameter Line Panel

Execute Dialogs

After selection the desired operation on the FCU Main window, the Execute button at the bottom of the window is enabled. Click **Execute** to start the selected operation. (If the **Execute** button is not enabled, you have not saved the current parameter set.)

WARNING: In some early versions of FCU for Windows 2000/2003/Windows NT systems, FCU may overwrite existing Windows 2000/2003/Windows NT systems target files without requesting confirmation.

If you started an FXmto operation and the target file already exists, FCU requests overwrite confirmation. Click **OK** to overwrite the target file, or click **Cancel** to cancel the operation.

When FCU starts the operation, the Execute panel opens and displays the progress of the operation. ***Note:*** The Execute dialog panel will not appear when the mainframe OS is VSE.

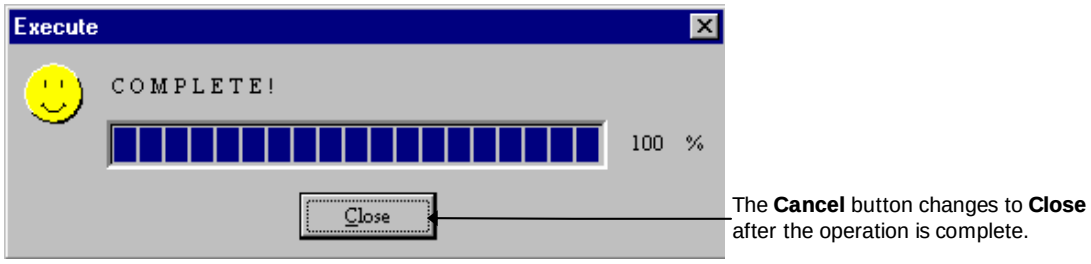


Figure 4-13 Execute Panel Showing Normal End

When the operation is complete, the Execute panel displays the result. If an error occurred, the Error information panel opens automatically to display the error. See Error Codes and Messages for further information on errors.

To cancel the operation in progress, select **Cancel**.

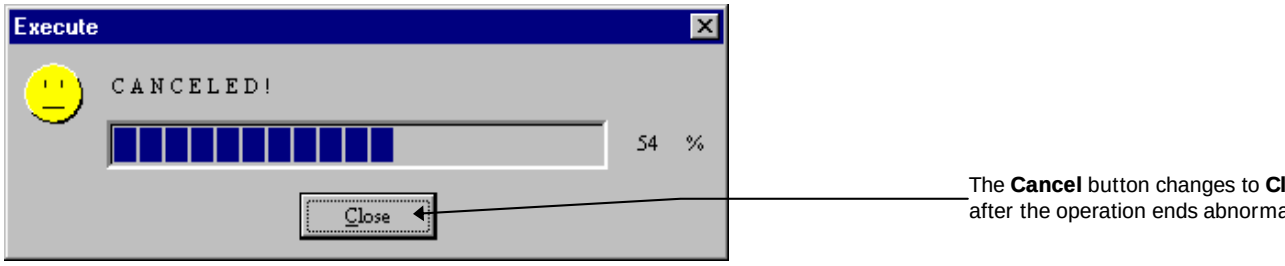


Figure 4-14 Execute Panel Showing Canceled Operation

If an error occurred, the Error information panel opens automatically to display the error). If there is an error during execution the panel below will be displayed. You should check that your parameters hae been entered correctly and try executing the command again. See Appendix C for further information on errors.

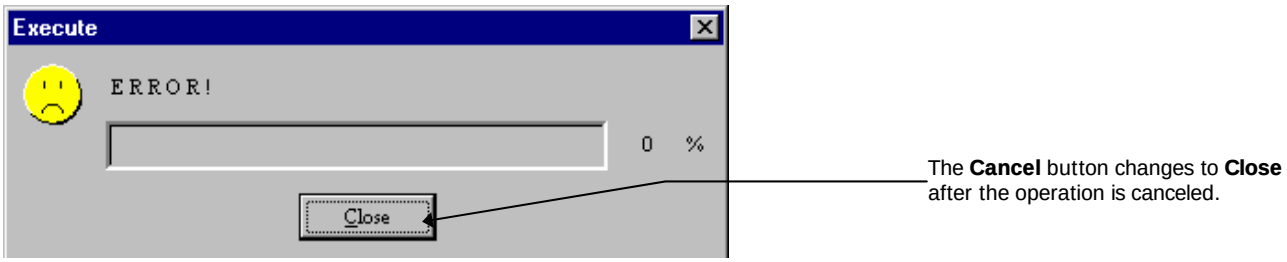


Figure 4-15 Execute Panel Showing Error End

Note: FCU does not load the next operation automatically. To perform another FX operation, select the desired **Parameter-Load** command, and repeat steps (8) through (12) as shown in Section File Conversion Utility Window. To exit FCU, select the **File-Exit** command.

Error Information Dialog

The **View-Error information...** command opens the Error Information panel, which displays the most recent error information (error code and message) for FCU, FAL, and system errors.

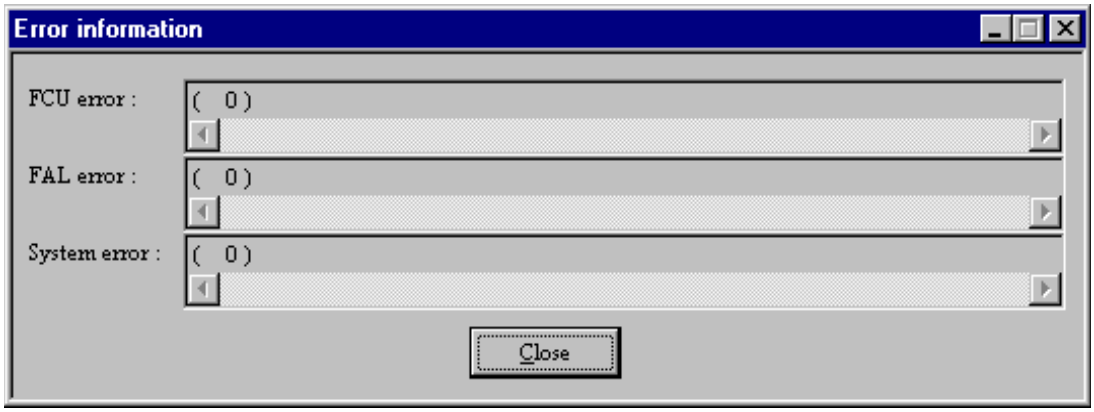


Figure 4-16 Error Information Panel

Log Files

The **View-Log file** command opens the log file for the current FCU parameter definition file using the Windows Notepad text editor. The log file contains the parameter sets executed, the date and time of execution, the result of each operation, and the error information (FCU, FAL, and Sys error codes) for each operation.

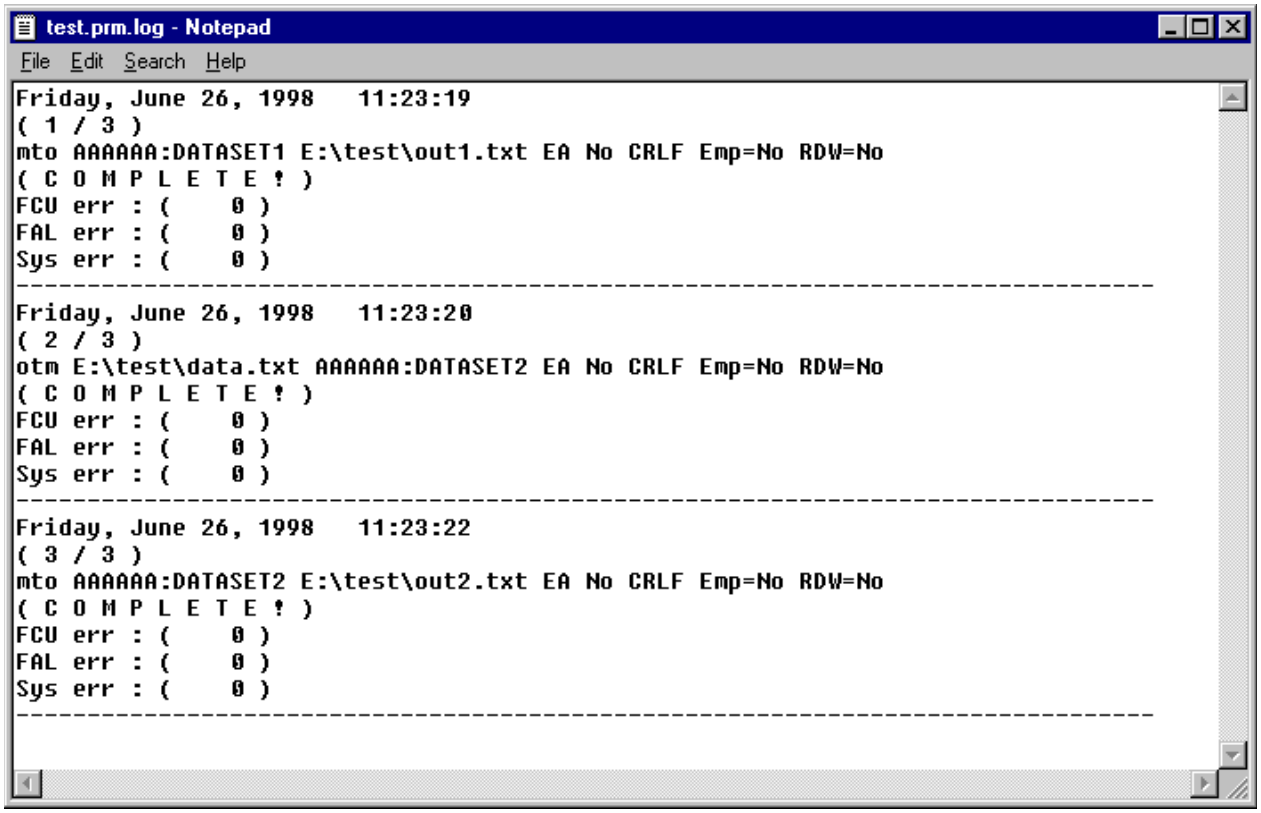

test.prm.log - Notepad
File Edit Search Help
Friday, June 26, 1998 11:23:19
(1 / 3)
mto AAAAAA:DATASET1 E:\test\out1.txt EA No CRLF Emp=No RDW=No
(C O M P L E T E !)
FCU err : (0)
FAL err : (0)
Sys err : (0)
Friday, June 26, 1998 11:23:20
(2 / 3)
otm E:\test\data.txt AAAAAA:DATASET2 EA No CRLF Emp=No RDW=No
(C O M P L E T E !)
FCU err : (0)
FAL err : (0)
Sys err : (0)
Friday, June 26, 1998 11:23:22
(3 / 3)
mto AAAAAA:DATASET2 E:\test\out2.txt EA No CRLF Emp=No RDW=No
(C O M P L E T E !)
FCU err : (0)
FAL err : (0)
Sys err : (0)

Figure 4-17 FCU Log File Display in Notepad

Format Utility for Windows

After the FAL/FCU software has been installed on the open-system host(s), you can format the FXoto volumes using the FX Formatter (FMT) utility. This enables you to allocate FXoto intermediate datasets. The FMT utility for UNIX is a UNIX command executed from the UNIX command line. The FMT utility for Windows 2000/2003/Windows NT is a GUI.

Note: FCU version 01-01-42 or later is required for the FMT utility.

The FX FMT utility defines the size of the OPEN-*x* volume in cylinders. The maximum number of cylinders allowed by FMT is shown in Table 3-8.

Table 4-1 FMT Utility Values

Emulation Type	01-XX-47 or earlier (*Note 2*)	01-XX-YY/ZZ not LUSE (*Note 2*)	01-XX-YY/ZZ LUSE (*Note 2*)	01-XX-YY/2x not LUSE (*Note 2*)	01-XX-YY/2x LUSE (*Note 2*) n=number of volumes (*Note 3*)
OPEN-3	3331	3331	5818	3331	(min 3338*n-7, 65534)

OPEN-8	5818	5818	5818	9959	(min 9996*n-7, 65534)
OPEN-9	5818	5818	5818	10009	(min 10016*n-7, 65534)
OPEN-E	not supported	5818	5818	19752	(min 19759*n-7, 65534)
OPEN-L	not supported	5818	5818	49429	(min 49439*n-7, 65534)
OPEN-V	not supported	not supported	not supported	65534	(min(Vc*n*128/ 96-7,65534) (see ***Note 4***)

Note 2: XX = 1 or 2; YY,x = number; ZZ<2*x*

Note 3: For Solaris, the data cylinder must be less than or equal to 32767. When using a LUSE volume, the geometry parameter is different, so the number of cylinders should be calculated as follows:

Cylinder (specified to FAL formatting) <= (A*B*C) / (15*96) – 5

A: Head (Geometry parameter)

B: Block/Track (Geometry parameter)

C: cylinder (Geometry parameter)

Note 4: Vc = OPEN-V cylinder value (MAX.:49160 cylinders using FAL)

The FMT utility can be used on standard-size OPEN-*x* volumes and on Virtual LVI/LUN (VIR) volumes.

Note: The VIR OPEN-*x* devices can also be called custom volume size (CVS) devices (e.g., OPEN-3-CVS). When formatting a VIR OPEN-*x* LU, use the number of cylinders defined for VIR minus seven (e.g., use 993 cylinders for a VIR device defined with 1000 cylinders). The cylinder size is: one cylinder = 15 tracks, one track = 96 sub-blocks, one sub-block = 512 bytes. Table 3-9 shows the relation between block length and write available capacity per track. The actual data capacity per cylinder = (write available capacity per track) × (15 tracks).

Table 4-2 Relation between Block Length and Write Available Capacity per Track

Block Length by Allocater = (A) (Bytes)	Write Available Data per Track (Bytes)	Block Length by Allocater = (A) (Bytes)	Write Available Data Per Track (Bytes)	Block Length by Allocater = (A) (Bytes)	Write Available Data Per Track (Bytes)
23477 - 32760	(A) × 1	1589 - 1684	(A) × 22	565 - 596	(A) × 44
15477 - 23476	(A) × 2	1493 - 1588	(A) × 23	533 - 564	(A) × 45
11477 - 15476	(A) × 3	1397 - 1492	(A) × 24	501 - 532	(A) × 46
9077 - 11476	(A) × 4	1333 - 1396	(A) × 25	469 - 500	(A) × 47

7477 - 9076	(A) × 5	1269 - 1332	(A) × 26	437 - 468	(A) × 48
6357 - 7476	(A) × 6	1205 - 1268	(A) × 27	405 - 436	(A) × 49
5493 - 6356	(A) × 7	1141 - 1204	(A) × 28	373 - 404	(A) × 50
4821 - 5492	(A) × 8	1077 - 1140	(A) × 29	341 - 372	(A) × 51
4277 - 4820	(A) × 9	1045 - 1076	(A) × 30	309 -340	(A) × 52
3861 – 4276	(A) × 10	981 - 1044	(A) × 31	277 - 308	(A) × 53
3477 –3860	(A) × 11	949 - 980	(A) × 32	245 - 276	(A) × 54
3189 – 3476	(A) × 12	917 - 948	(A) × 33	213 - 244	(A) × 55
2933 – 3188	(A) × 13	853 - 916	(A) × 34	181 - 212	(A) × 56
2677 – 2932	(A) × 14	821 - 852	(A) × 35	149 - 180	(A) × 57
2485 – 2676	(A) × 15	789 - 820	(A) × 36	117 - 148	(A) × 58
2325 – 2484	(A) × 16	757 - 788	(A) × 37	85 - 116	(A) × 59
2165 – 2324	(A) × 17	725 - 756	(A) × 38	53 - 84	(A) × 60
2005 – 2164	(A) × 18	693 - 724	(A) × 39	21 - 52	(A) × 61
1877 – 2004	(A) × 19	661 - 692	(A) × 40	1 - 20	(A) × 62
1781 – 1876	(A) × 20	629 - 660	(A) × 41	—	—
1685 – 1780	(A) × 21	597 - 628	(A) × 42	—	—

Note: The write available data per track includes the four-byte RL information and four-byte BL information for each record. When transferring variable-length records, make sure to take this extra required space into account.

WARNING: The FX FMT utility erases all data on the OPEN-*x* LU being formatted. If necessary, back up the data on the OPEN-*x* LUs prior to FMT formatting.

To format an OPEN-x volume using the FX FMT utility for Windows 2000/2003/ Windows NT:

1. Log in to the system as administrator.
2. Double-click on the **Format** icon to open the Format panel (see Figure 3-1).

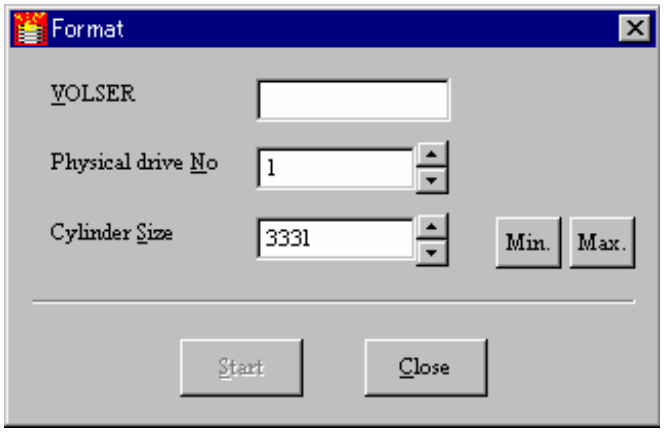


Figure 4-18 FMT Utility for Windows 2000/2003/Windows NT

3. On the Format panel, enter the six-character volume serial number for the OPEN-*x* volume being formatted in the **VOLSER** field. Make sure to use the same volser for this volume in the FXoto volume definition file.
4. Specify the physical drive number (device number) for the OPEN-*x* volume being formatted in the **Physical drive No** field. Make sure to use the same physical drive number for this volume in the FXoto volume definition file.
5. Specify the number of cylinders for the OPEN-*x* volume in the **Cylinder Size** field. The **Min.** button enters 2 (two) cylinders, and the **Max.** button enters 5818 cylinders. If the OPEN-*x* volume is standard size (e.g., OPEN-3), use the maximum size of 5818 cylinders. If the OPEN-*x* volume is custom size (e.g., OPEN-3*n VIR device), use the following value: (# of cylinders defined for VIR) – 7. For example, if the VIR OPEN-*x* volume is defined with 1000 cylinders, enter 993 in the **Cylinder Size** field.

   Note: The maximum size for the Allocater is 4369 cylinders.
6. When the VSN, physical drive number, and cylinder size are correct, select the **Start** button. When the Format confirmation appears (see Figure 3-2), select **OK** to perform the requested FMT format operation, or select **Cancel** to cancel your request.

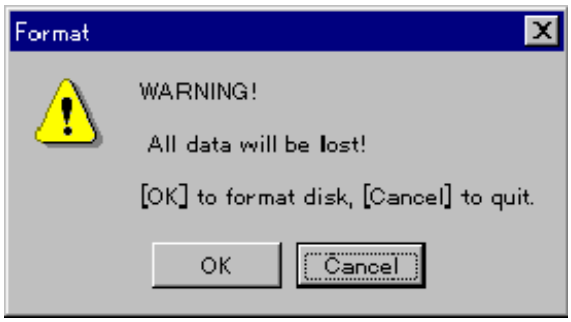


Figure 4-19 FMT Format Warning and Confirmation

7. When the format operation completes successfully, the **Format complete** message is displayed (see Figure 3-3).

Figure 4-20 FMT Format Complete Message

If the specified volser has already been used, an error message is displayed to notify you (see Figure 3-4).

Figure 4-21 FMT VOLSER Used Message

If the format operation could not be started due to an error condition, the **Format check error** message is displayed.

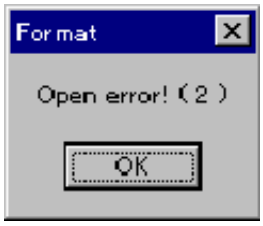


Figure 4-22 FMT Error Message

If the format operation did not complete successfully, one of the following error messages is displayed (see Figure 3-5) (n = system error code):

Open error! (n)	Open process error on specified volume.
Seek error! (n)	Seek process error on specified volume.
Read error! (n)	Read process error on specified volume.
Write error! (n)	Write process error on specified volume.
Close error! (n)	Close process error on specified volume.

8. When you are finished formatting OPEN-*x* volumes for use as FXoto volumes, select **Close** to close the Format panel and exit the FMT utility.

Important Note: When the message "**Format check error**" is indicated, the formatting operation has not started and the original condition has been kept. When another message is indicated, the formatting process has already started. The data on the volume has already been initialized. Remove the error condition and format the volume again.

If you execute over the maximum number of cylinders on Windows 2003, the message "Format failed … Format check error" appears after about a minute. It should be clear when formatting has completed.

Allocation Utility for Windows

When you perform FXoto operations using OPEN-*x* FMT volumes, you must allocate the intermediate datasets before starting the file transfer operations. The FX Allocater (ALC) utility can only be used on OPEN-*x* volumes which have already been formatted using the FX FMT utility (see section Format Utility for Windows).

Note: FCU version 01-01-41 or later is required for the ALC utility.

Note: For versions 01-01-41: The ALC utility for UNIX is a UNIX command executed from the UNIX command line. The ALC utility for Windows 2000/2003/Windows NT systems is a GUI. The ALC utility for UNIX can only be used on volumes formatted with the FMT utility for UNIX. The ALC utility for Windows 2000/2003/Windows NT systems can only be used on volumes formatted with the FMT utility for Windows 2000/2003/Windows NT systems.

CAUTION: The capacity of the intermediate dataset varies depending on block length. Remember to calculate the required size for the intermediate dataset. When you transfer variable-length records, make sure to take the four-byte RL information and four-byte BL information for each record into account.

To allocate an intermediate FXoto dataset using the ALC utility:

1. Log in to the system as administrator.
2. Double-click on the **Allocate** icon to start the ALC utility and open the Allocation panel.

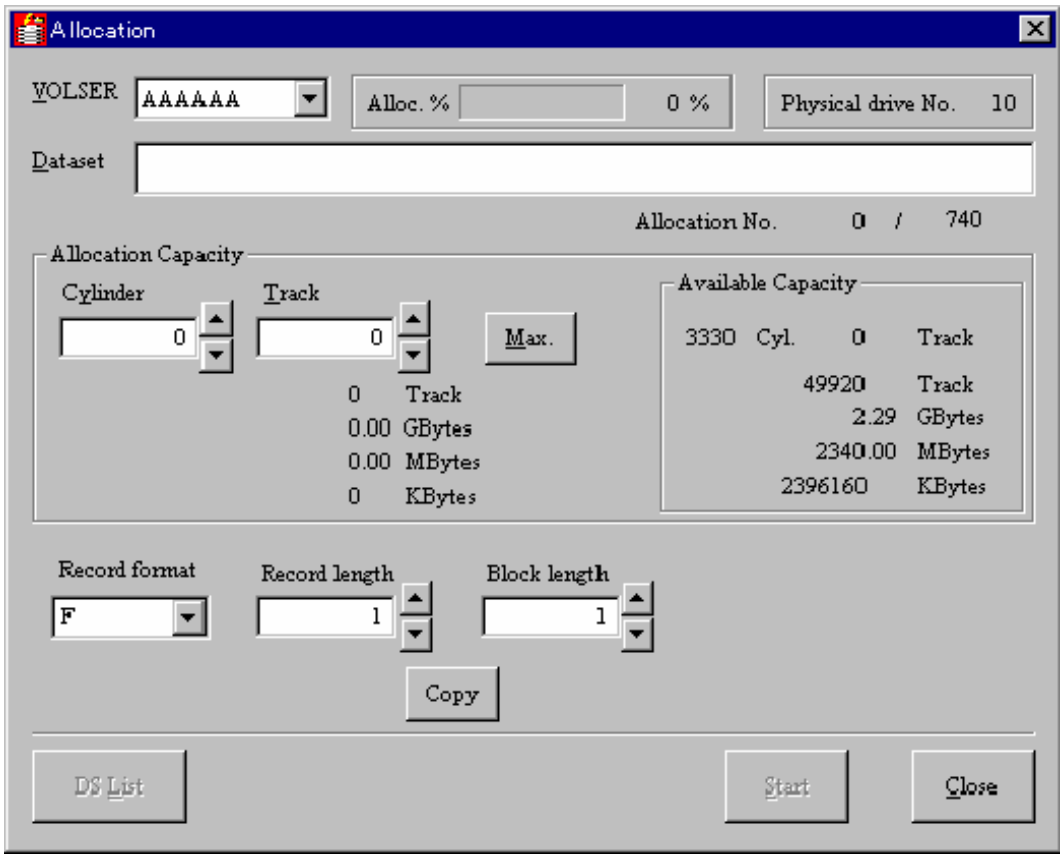


Figure 4-23 ALC Utility for Windows 2000/2003/Windows NT systems

3. The ALC utility automatically displays the first OPEN-*x* FMT volume (in alphanumeric order) in the **VOLSER** field. If this is not the desired volume, select the desired volume from the drop-down list of volsers. If ALC could not find any OPEN-*x* FMT volumes, ALC displays the **FX format disk not found** message.

Figure 4-24 ALC Disk Not Found Message

4. Enter the name of the dataset being allocated in the **Dataset** field (maximum forty-four characters: A-Z, 0-9, @, #, ., \). Do not use any spaces or symbols other than @, #, ., and \.

5. Enter or select the size of the new dataset (number of cylinders, number of tracks) in the **Cylinder** and **Track** fields. The file size will be (# of cyl) + (# of tracks). The **Max.** button enters the maximum size for the new dataset in the **Cylinder** and **Track** fields based on the available capacity. The **Available Capacity** box displays the free space on the specified volume, so that you can select the appropriate size for the new dataset.
6. Enter or select the record format in the **Record format** field: F, FB, V, or VB.
7. Enter or select the record length in the **Record length** field:
   - When record format = **F**, record length = block length.
   - When record format = **FB**, record length = block length ÷ N (N = integer).
   - When record format = **V or VB**, 5 ≤ record length ≤ (block length – 4).
8. Enter or select the block length in the **Block length** field. If block length = record length, select the **Copy** button to copy the record length into the **Block length** field.
   - When record format = **F or FB**, 1 ≤ block length ≤ 32760.
   - When record format = **V or VB**, 9 ≤ block length ≤ 32760.
9. When all parameters for the new dataset are correct, select the **Start** button.
10. When the allocate operation completes successfully, the **Allocation complete** message is displayed.

Figure 4-25 ALC Allocation Complete Message

If the allocate operation could not be started due to an error condition, the **Allocate check error** message is displayed.

Figure 4-26 ALC Error Message

If the allocate operation did not complete successfully, one of the following error messages is displayed (n = system error code):

Open error! (n)	Open process error on the FXoto volume.
Seek error! (n)	Seek process error on the FXoto volume.
Read error! (n)	Read process error on the FXoto volume.
Write error! (n)	Write process error on the FXoto volume.
Close error! (n)	Close process error on the FXoto volume.

11. When you are finished allocating datasets on FXoto volumes, select **Close** to close the Allocation panel and exit the ALC utility.

5

Performing Cross-OS File Exchange Operations

This chapter describes how to operate FX.

- Performing File Transfer Operations - UNIX
- Performing File Transfer Operations – Windows

FX file transfer operations are performed using the FCU GUI software installed on the open-system host(s) attached to the TagmaStore USP and NSC subsystem.

The FCU GUI enables you to perform file transfer operations interactively, provides access to detailed information on the FX source datasets and files, and displays error information for FX operations. The FCU GUI also allows you to create and modify FCU parameter definition files interactively.

When you perform FX operations that access datasets on -A, -B, or -C FX volumes, FCU must have access to the FX volume definition file that defines these volumes. When you perform FX operations that access ALC-generated datasets on OPEN-*x* FMT FX volumes, FCU must have access to the separate FXoto volume definition file that defines the OPEN-*x* FMT volumes. Since FCU can only access one FX volume definition file at a time, the FCU parameter definition files must also keep operations using OPEN-*x* FMT volumes separate from operations using -A, -B, or -C FX volumes. Before you start FCU GUI operations, make sure that the desired FX volume definition file is available (**datasetmount.dat** in current directory) and that the desired FCU parameter definition file contains FX operations which access the volumes defined in the FX volume definition file. FCU will not be able to perform operations which access volumes that are not defined in the current FX volume definition file.

The FCU GUI for UNIX -based platforms and the FCU GUI for Windows 2000/2003/Windows NT systems are significantly different. Section Performing File Transfer Operations - UNIX describes and provides instructions for using the FCU GUI for UNIX. Section Performing File Transfer Operations – Windows describes and provides instructions for using the FCU GUI for Windows 2000/2003/Windows NT systems. For information on using FCU from the UNIX command line (without the GUI), see Appendix B.

For information on using the FAL C functions (Visual C++® for Windows 2000/2003/Windows NT systems), which enable user programs on the open-system host to access z/OS, datasets on FX volumes, see 0.

Performing File Transfer Operations - UNIX

The FCU GUI enables you to perform FX file transfer operations interactively and provides access to detailed information on the datasets/files in the specified FX source volume/directory. The FCU GUI displays the FX operations in the FCU parameter definition file (if specified), allows you to modify the FCU parameter definition file interactively, and also allows you to enter FCU parameters and perform FX operations manually. The FCU GUI also displays the error information for FX operations.

Starting the FCU GUI for UNIX

To start the FCU GUI program for UNIX -based platforms:

1. At the UNIX command line prompt, enter: **fcu [-nc] [param]**

   The **-nc** option (**nc** = no checking) tells FCU to execute all specified FX operations without requesting confirmation for FCU parameters or checking for existing FXmto target files. If you want to bypass these confirmations, enter **-nc**.

   The **param** option tells FCU whether to use the FCU parameter definition file or a specific FCU initiation parameter set to perform FX operations. The **param** option must have one of the following three values:

   – [blank]. If you want to use the default FCU parameter definition file (**fcudata.param** in the current directory), leave the **param** option blank (do not enter anything).

   – **file_name**. If you want to use a different FCU parameter definition file, enter the file name with complete path (absolute or relative) if not in the current directory.

   – **-P + parameters**. If you want to perform one specific FX operation, enter **-P** followed by the FCU initiation parameter set (e.g., **mto VSN:dataset targetfile No No No**) for the desired FX operation. The **-P** option requires the **-nc** option.

   For example:

   – If you want to use the default FCU parameter definition file and check the parameters and FXmto target files, enter: **fcu**

   – If you want to use the default FCU parameter definition file and perform all operations without checking parameters or FXmto target files, enter: **fcu -nc**

   – If you want to use a different FCU parameter definition file and perform all operations without checking parameters or FXmto target files, enter: **fcu -nc filename**

   – If you want to perform one specific FX operation, enter: **fcu -nc -P [parameters]**

Note: The following warnings may appear during FCU startup. These warnings do not affect FCU and can be ignored.

WARNING: Missing characters in String to FontSet conversion.

WARNING: Cannot convert string “-dt-interface system-medium-r-normal -m*-*-*-*-*-*-*-*-*” to type FontSet.

2. The FCU GUI program now starts loading. The FCU version and copyright screen (see Figure 5-1) is displayed while FCU is loading. When FCU is finished loading, the FCU main panel is displayed.
3. If you specified the **-nc** option, FCU processes all specified operations, overwrites existing mto target files, terminates, and displays any error information at the UNIX prompt.

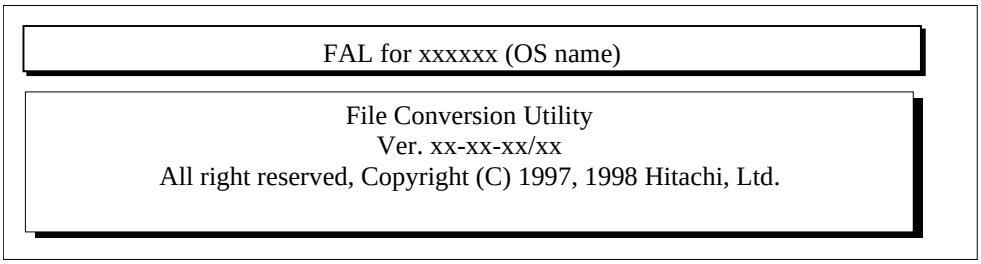


Figure 5-1 FCU Version and Copyright Screen (UNIX)

Performing File Transfer Operations (UNIX)

To perform file transfer operations using the FCU GUI for UNIX:

1. If you will be performing FXmto operations:
   a) Make sure that the source datasets are located on the desired FX volume(s). If you will not be using an existing FCU parameter definition file, write down the VSN:dataset of the source dataset and the complete path and file name of the target file for each FXmto operation.
   b) Verify that the FXmto target files do not already exist (or can be overwritten).
   c) Vary the FXmto volume(s) and channel path(s) offline from the z/OS host.
2. If you will be performing FXotm operations:
   a) Make sure that the source files are located on the desired FX volume(s). If you will not be using an existing FCU parameter definition file, write down the complete path and file name of the source file and the VSN: dataset of the target dataset for each FXotm operation.
   b) Create and allocate the target datasets. This ensures that the target dataset is registered in the VTOC. Make sure to allocate enough space and to use the appropriate record format and record length for the data to be transferred.
   c) Vary the FXotm volume(s) and channel path(s) offline from the z/OS host.
3. If you will be performing FXoto operations:
   a) If you will not be using an existing FCU parameter def. file, write down the complete path and file name of the source and target files for each FXotm/mto operation.
   b) Allocate the intermediate datasets on the FXoto volume(s). Use the ALC utility on OPEN-x FMT volumes. Make sure to allocate enough space and to use the appropriate record format and record length for the data to be transferred.
   c) Verify that the FXoto target files do not already exist (or can be overwritten).
4. Make sure that the desired FX volume definition file (FXoto only, or FXmto and FXotm) is available for use by FCU (**datasetmount.dat** in current directory).
5. Start FCU with the desired options (see section Starting the FCU GUI for UNIX).

   Note: If you specify the **-nc** option, FCU performs all specified operations continuously, then self-terminates and displays any error information at the UNIX prompt.

6. When the FCU main panel opens, make sure that the **Volume File** field displays the FX volume definition file (**datasetmount.dat**). If the FX volume definition file is not displayed (or if incorrect), FCU will not be able to perform FX operations.
7. Make sure that the **Parameter File** field displays the desired FCU parameter definition file. If not, enter the desired FCU parameter definition file name (with complete path if not in the current directory), and select the **File-Load** command to open the file. If you want to create a new file using the FCU GUI, see section Using the `listvol` Function (UNIX) for instructions.
8. The FCU main panel displays the first/next parameter set in the specified FCU parameter definition file. If you want to perform this FX operation, click **OK**. If not:
   a) You can load the next parameter set using the **File-Load** command.
   b) You can delete the current parameter set from the FCU parameter definition file using the **File-Delete** command. The next parameter set loads automatically.
   c) You can modify the current parameter set as follows: Change the FCU parameters as needed, and then use the **File-Save** command to save your changes in the FCU parameter definition file (replaces the previously loaded parameter set).
   d) You can add a new parameter set to the end of the file as follows: Select **File-Load** until you reach the end of the file, enter the desired parameters, and then select **File-Save** to add the new line at the end. If you want to insert a new line between existing lines, edit the FCU parameter definition file later using a text editor.
9. When the desired FX operation is displayed, click **OK** to start the operation. (If the **OK** button is not enabled, you have not saved the current parameter set.)
10. If you started an FXmto operation and the target file already exists, FCU requests overwrite confirmation. Click **OK** to overwrite the target file, or click **Cancel** to cancel the operation.
11. When FCU starts the operation, the **Status** field displays the progress of the operation. If desired, while the operation is in progress, you can load another parameter set and click **OK** to start the next operation right after the current operation completes.

    Note: Be careful when doing this. If you click buttons or menu commands while an operation is in progress, FCU will save and execute those commands when the current operation completes.
12. When the FX operation is complete, the **Status** field displays **Complete**. If an error occurred, the error information display opens automatically to display the error. See Appendix C for further information on error conditions.

13. FCU does not load the next operation automatically. To perform another FX operation, select **File-Load**, and repeat steps (8) through (12). To exit FCU, select the **File-Exit** command.

Using the `listvol` Function (UNIX)

The **listvol VSN** function enables FCU users to access the z/OS dataset information without having to launch the FCU GUI (and use the **Help-MF-File** command). The **listvol VSN** UNIX command displays the dataset information for the specified VSN. The **listvol VSN** function requires the FX volume definition file.

Note: The **listvol VSN** function is available in FCU for UNIX versions 01-01-41 and later.

```
# listvol volser                                          ← Specify 6-character VSN.

  Dataset Name      DO     RF    RL     BL     TT    R     EX (Cyl:Trk)
_____________________________________________________________________
*SAMFILE01.FIX      SAM    F     4096   4096   1     10    100:0
-DAMFILE.F          DAM    F     128    4096   0     10    100:0
?SAMFILE.VSE        SAM    ?     0      0      0     0     0:0

0                                                       ← Return value (normal end).
#
```

Figure 5-2 Listvol VSN Function

The **listvol VSN** function displays the following information:

- **Dataset name**. An asterisk (*) before the dataset name indicates that FAL/FCU can process the dataset. A dash (-) indicates that FAL/FCU cannot process the dataset. A question mark (?) before the dataset name indicates that FCU can process the dataset only if the VSE record option is used to specify the RF, RL, and BL.
- Dataset organization (**DO**) type: SAM, DAM, PAM, VSAM, ??? = unknown. FAL/FCU supports SAM datasets.
- Record format (**RF**): F = fixed length, V = variable length, U = undefined length, S = spanned record, ? = unknown. FAL/FCU supports F and V record formats.
- Record length (**RL**): in bytes
- Block length (**BL**): in bytes
- **TT**+**R**: last block address
- **EX (Cyl:Trk)**: data extent size (number of cylinders:number of tracks)

- Return value: **0** indicates normal end; **1** indicates error end. If an error occurred, the error code and message are displayed and the error is logged in the error log file.

Creating FCU Parameter Definition Files (UNIX)

To create an FCU parameter definition file using the FCU GUI for UNIX:

1. Start the FCU GUI for UNIX by entering **fcu** (see section Starting the FCU GUI for UNIX). Do not specify the **-nw**, **-nc**, or **param** option.
2. When the FCU main panel opens (see section Performing File Transfer Operations (UNIX)), enter the desired file name in the **Parameter File** field (with complete path if you do not want to save the file in the current directory).
3. If you plan to perform FX operations while you are creating the FCU parameter definition file, make sure that the **Volume File** field displays the correct FX volume definition file (**datasetmount.dat**). If not (or if incorrect), FCU will not be able to perform FX operations, but you can still create a new FCU parameter definition file.
4. Select the **File-Load** command to open the new file.
5. Enter the desired FCU initiation parameters for the first FX operation:
   - Select the file transfer direction using the **M to O** button or **O to M** button.
   - Enter the source and target files in the **Input File** and **Output File** fields (**VSN:dataset**, **filename** with complete path if not in current directory).
   - Select the desired FCU file transfer options: **Code Conversion**, **Padding**, **Delimiter**, **Emp**, **RDW**, and VSE. See section Performing File Transfer Operations (UNIX) for further information on these options.
6. When the FCU initiation parameters are correct, select the **File-Save** command to add this parameter set as the first line in the new FCU parameter definition file. If the FX volume definition file is correct, you can perform the operation now by clicking **OK**. If the **OK** button is not enabled, the parameter set has not been saved in the file.
7. Select the **File-Load** command to load the next line. The **Status** field should indicate that you are at the end of the file. The FCU GUI for UNIX only allows you to add new lines when you are at the end of the file (right after the last line).
8. Repeat steps (5), (6), and (7) to add each parameter set to the new FCU parameter definition file. Make sure to keep FX operations which use OPEN-*x* FMT volumes in a separate FCU parameter definition file from operations which use -A, -B, -C volumes.

9. If you need to modify an existing line, go to the line to be modified using the **File-Load** command, change the parameters as needed, and then use the **File-Save** command to replace the line that was loaded.
10. If you need to insert a new line between existing lines, use a text editor later to edit the file. You cannot add a new line between existing lines using the FCU GUI for UNIX.
11. When you are finished adding lines to your new FCU parameter definition file, make sure that you have selected the **File-Save** command for the last parameter set you added or modified, and then select the **File-Exit** menu command to close the file and exit FCU.

Creating Multiple Volume Definition Files (UNIX)

Multiple volume Dataset is supported for FXmto, when the version is 01-*XX*-50/YY (*XX*=01 or 02) or later. Multiple volume Dataset is also supported for FXotm when the version is 01-*XX*-60/YY (*XX*=02, 03 or 04) or later. The name of the multiple volume definition file is ***"multidef.dat***". It must be placed under the current directory where the FAL/FCU is to be executed. Specify the other name in "FAL_MULTI_DEF_FILE" of the environment variable, if the name of the multiple volume definition file want to be changed.

Note: Multiple volume Dataset is supported in the case of FileExchangeotm when the version is 01-*XX*-60/YY(*XX*=02, 03 or 04) or higher.

Table 5-1 illustrates the VSN function.

Table 5-1 VSN Function

VSN:DSN[,VOLID1]	VSN[,VOLID2]	-----	VSN[,VOLIDn]	
i)	ii)		iii)	iv)
end				
v)				

e. i) This parameter is the information of the head volume:

- VSN: a volume serial number with six digit of alphabet (A-Z, @, #, and \) or numeral (0-9) characters.
- DSN: dataset name.(Use maximum 44-digit of alphabet or numeral character)
- VOLID1: a VSN identification (Omit this parameter if a VSN identification is omitted in the volume definition file, and specify same as the volume definition file, if a VSN identification is specified in the volume definition file.)

f. ii) This parameter is the information of the second volume:

- VSN: a volume serial number with six digit of alphabet (A-Z, @, #, and \) or numeral (0-9) characters.

- VOLID2: a VSN identification (Omit this parameter if a VSN identification is omitted in the volume definition file, and specify same as the volume definition file, if a VSN identification is specified in the volume definition file.)

g. iii) This parameter is the information of the last volume.(The number of volume is ‘n’):

- VSN: a volume serial number with six digit of alphabet (A-Z, @, #, and \) or numeral (0-9) characters.
- VOLIDn: a VSN identification (Omit this parameter if a VSN identification is omitted in the volume definition file, and specify same as the volume definition file, if a VSN identification is specified in the volume definition file.)

h. iv) Each line above must be separated by using ***“Return”*** key.

i. v) The ***“end”*** specifies that the volume definition file ends here.

Note: Each parameter must be separated with one or more ***"space"*** character(s).

Note: One data set information must be specified in one line.

Note: 999 information can be specified in the multiple volume definition file.

Using FCU from the Command Line (UNIX)

FCU can be used without the GUI to perform FX operations. To use FCU without the GUI, you must start FCU using the **-nw** option. The FCU options are:

- The **-nw** option (**nw** = no Windows) tells FCU to execute the specified FX operation(s) without displaying the GUI. When you use this option (entered as **fcunw** or **fcu -nw**), FCU uses the FCU initiation parameters specified by the **param** option to perform FX operations. To use FCU without the GUI, the **-nw** option is required.

  Note: FCU versions 01-01-36 and later for Windows 2000/2003/Windows NT systems do not support the **-nw** option.

- The **-nc** option (**nc** = no checking) tells FCU to execute all specified FX operations without requesting confirmation for FCU parameters or existing FXmto target files. If you want to bypass these confirmations, enter **-nc**. FCU will perform all specified operations and overwrite existing FXmto target files. If you want to check the FCU parameters and the FXmto target file before starting each operation, do not enter **-nc**.

- The **param** option (param = FCU parameter definition file) tells FCU whether to use an FCU parameter definition file or a specific FCU initiation parameter set to perform FX operations. The **param** option must have one of the following three values:

  – [blank]. If you want to use the default FCU parameter definition file (**fcudata.param** located in the current directory) to perform FX operations, leave the **param** option blank (do not enter anything).

  – **file_name**. If you want to use a different FCU parameter definition file to perform FX operations, enter the filename of the file. Make sure to enter the complete path (absolute or relative path) if the file is not in the current directory.

  – **-P + parameters**. If you want to perform one specific FX operation, enter **-P** followed by the FCU initiation parameter set (e.g., **mto VSN:dataset targetfile No No No**) for the desired FX operation.

Note: FCU for UNIX cannot be used by a "signal handler." If this accidentally happens and memory space is occupied, use **kill** to cancel the processes, and use **ipcrm** to delete the shared memory areas that have KEY=0 (refer to OS manuals). Do not issue the following signals to an FCU process (UNIX only):

SIGUSR1, SIGUSR2, SIGILL, SIGTRAP, SIGIOT, SIGABRT, SIGEMT, SIGFPE, SIGKILL, SIGBUS, SIGSEGV, SIGSYS, SIGALRM, SIGPOLL, SIGIO, SIGSTOP, SIGTSTP, SIGCONT, SIGTTIN, SIGTTOU, SIGVTALRM, SIGPROF, SIGXCPU, SIGXFSZ, SIGWAITING, SIGLWP, SIGFREEZE, SIGTHAW, SIGCANCEL

To perform FX operations using FCU without the GUI:

1. If you will be using an FCU parameter definition file to perform FX operations, make sure that the file contains the correct FCU initiation parameter sets for the FX operations you want to perform. If you will not be using the default FCU parameter definition file, note the name and location of the file.
2. Log in as root on the UNIX server, and enter: **fcunw [-nc] [param]**
   For example (see Figures Figure 5-3, Figure 5-4, and Figure 5-5):
   - To perform the FX operations in the default FCU parameter definition file with confirmations, enter: **fcunw**
   - To perform the FX operations in the default FCU parameter definition file without confirmations, enter: **fcunw -nc**
   - To perform the FX operations in a different FCU parameter definition file with confirmations, enter: **fcunw /directory/directory/file_name**
   - To perform the FX operations in a different FCU parameter definition file without confirmations, enter: **fcunw -nc /directory/directory/file_name**
   - To perform one specific FX operation with confirmations, enter: **fcunw -P mto VSN:dataset targetfile No No No**
   - To perform one specific FX operation without confirmations, enter: **fcunw -nc -P mto VSN:dataset targetfile No No No**
3. If you specified the **-nc** option, FCU will perform all specified FX operations without requesting confirmation for the FCU parameters or for existing FXmto target files.
   - If you did not specify the **-nc** option, FCU will display the FCU initiation parameters for the operation to be performed and request confirmation. Enter **ok** to perform the specified FX operation, or enter **cancel** to load the next set of FCU parameters.
   - If you did not specify the **-nc** option and the FXmto target file already exists, FCU will request confirmation to overwrite the file. Enter **ok** to overwrite the existing file, or enter **cancel** to load the next set of FCU initiation parameters.
4. When the FX operation starts, FCU displays **Start**. When the operation completes successfully, FCU displays **Complete**. If the operation does not start or complete successfully, FCU displays an error message and loads the next parameter set.
5. When the last FCU initiation parameter set is processed (or canceled by the user), the FCU program terminates and returns an ending status value. The ending status is included in **$status** for C-shell and **$?** for B-shell/K-shell.

   0 = successful completion. All FX operations completed successfully.
   1 = unsuccessful completion. One or more operations did not complete successfully.

```
# fcunw                                                  ← Start FCU with checking.
File Conversion Utility Ver.01-01-40/00                  ← FCU program version.

mto VSN:dataset file_name EA No LF                       ← First set of parameters.
ok/cancel ? ok                                           ← Enter ok or cancel.
Now checking...                                          ← Checking for target file.
Complete                                                 ← Operation completed.

otm file_name VSN:dataset EA No No                       ← Next set of parameters.
ok/cancel ? ok                                           ← Enter ok or cancel.
Input file : Open error (-350)                           ← Error info displayed.
  (Fal error    : xxx
  (System error : xxx

mto VSN:dataset file_name EA No LF                       ← Next set of parameters.
ok/cancel ? ok                                           ← Enter ok or cancel.
Now checking...                                          ← Checking for target file.
OverWrite ? ok/cancel ? ok                               ← Enter ok to overwrite file.
Complete                                                 ← Operation completed.

mto VSN:dataset file_name EA No LF                       ← Next set of parameters.
ok/cancel ? cancel                                       ← Enter ok or cancel.
 :
 :
#
```

Figure 5-3 Using FCU From the UNIX Command Line

```
# fcunw -nc       ← Start FCU without checking.
File Conversion Utility Ver.01-01-40/00                  ← FCU program version.

mto VSN:dataset file_name EA No LF                       ← First set of parameters.
Now checking...                                          ← Starting FX operation.
Complete                                                 ← Operation completed.

otm file_name VSN:dataset EA No No                       ← Next set of parameters.
Input file : Open error (-350)                           ← Error info. displayed.
  (Fal error    : xxx
  (System error : xxx

mto VSN:dataset file_name EA No LF                       ← Next set of parameters.
Now checking...                                          ← Starting FX operation.
Complete                                                 ← Operation completed.

mto VSN:dataset file_name EA No LF                       ← Next set of parameters.
 :
 :
#
```

Figure 5-4 Using the -nc Option

```
# fcunw -nc -P mto VSN:dataset file_name EA No LF        ← Start FCU without checking.

mto VSN:dataset file_name EA No LF                       ← Specified FCU parameters.
Now checking...                                          ← Starting FX operation.
Complete                                                 ← Operation completed.
#
```

Figure 5-5 Using the -P param Option

Performing File Transfer Operations – Windows

Starting the FCU GUI

To start the FCU GUI program for Windows 2000/2003/Windows NT systems:

1. Log on with Administrator access privileges.
2. Start the FCU GUI as follows: Click **Start-Programs-FCU-FCU**, or open the **c:** folder and double-click on **FCU**, or create a shortcut for **FCU** on the desktop.

   Note: Do not start FCU by dragging and dropping an FCU parameter definition file on the FCU program icon. FCU program operation cannot be guaranteed.
3. If you want to specify any of the FCU options, start FCU from the command line (DOS prompt) as follows: Go to the FCU directory (containing **fcu.exe** and **datasetmount.dat**), and enter **fcu [-nc] [-cl] [param]**

   The **-nc** option is the same as for UNIX: All specified FX operations are performed without confirmation of FCU parameters or FXmto target file overwrites.

   The **-cl** option specifies that all FCU log files will be cleared before starting.

   The **param** option is the same as for UNIX:

   – If you want to open a new untitled FCU parameter definition file when you start FCU, leave the **param** option blank.

   – If you want to load an FCU parameter definition file when you start FCU, enter the file name with complete path if the file is not in the current directory.
4. The FCU GUI program now starts loading. The FCU version and copyright screen (see Figure 5-6) is displayed while FCU is loading. When FCU is finished loading, the FCU main panel is displayed (see section Performing File Transfer Operations (UNIX)).
5. If you started FCU from the DOS prompt and specified the **-nc** option, FCU processes all specified operations, overwrites existing FXmto target files, and then terminates and displays any error information at the DOS prompt.

Figure 5-6 FCU Version and Copyright Screen (Windows 2000/2003/Windows NT Systems)

Performing File Transfer Operations (Windows)

To perform FX file transfer operations using the FCU GUI for Windows 2000/2003/Windows NT systems:

1. If you will be performing FXmto operations:
   a) Make sure that the source datasets are located on the desired FX volume(s). If you will not be using an existing FCU parameter definition file, write down the VSN:dataset of the source dataset and the complete path and file name of the target file for each FXmto operation.
   b) Verify that the FXmto target files do not already exist (or can be overwritten).
   c) Vary the FXmto volume(s) and channel path(s) offline from the z/OS host.
2. If you will be performing FXotm operations:
   a) Make sure that the source files are located on the desired FX volume(s). If you will not be using an existing FCU parameter definition file, write down the complete path and file name of the source file and the VSN:dataset of the target dataset for each FXotm operation.
   b) Create and allocate the target datasets. This ensures that the target dataset is registered in the VTOC. Make sure to allocate enough space and to use the appropriate record format and record length for the data to be transferred.
   c) Vary the FXotm volume(s) and channel path(s) offline from the z/OS host.
3. If you will be performing FXoto operations:
   a) If you will not be using an existing FCU parameter def. file, write down the complete path and file name of the source and target files for each FXotm/mto operation.
   b) Use the ALC utility to allocate the intermediate datasets on the FXoto volume(s). Make sure to allocate enough space and to use the appropriate record format and record length for the data to be transferred.
   c) Verify that the FXoto target files do not already exist (or can be overwritten).
4. Make sure that the desired FX volume definition file (FXoto only, or FXmto and FXotm) is available for use by FCU (**datasetmount.dat** in current directory).
5. Start FCU (see section Starting the FCU GUI). If you want to specify any FCU options, start FCU from the DOS prompt.

   Note: If you specify the **-nc** option, FCU performs all specified operations continuously, then self-terminates and displays any error information at the DOS prompt.

6. When the FCU main panel opens, select the **View-Volume information…** command ( ) to open the **Volume information** panel, and verify that the desired FX volume(s) is/are available. If not, edit the FX volume definition file as needed.
7. Make sure that the desired FCU parameter definition file is open (displayed in title bar). If not, open the desired FCU parameter definition file using the **File-Open** command ( ). If you want to create a new file using the FCU GUI, see section Creating FCU Parameter Definition Files (Windows) for instructions.

8. The FCU main panel displays the first/next parameter set in the specified FCU parameter definition file. If you want to perform this FX operation, click **Execute**. If not:

   a) You can load the next parameter set using the **Parameter-Load-Next** command ( ).

   b) You can delete the current parameter set from the FCU parameter definition file using the **Parameter-Delete** command ( ). The next parameter set loads automatically.

   c) You can modify the current parameter set as follows: Change the FCU parameters as needed, and then use the **Parameter-Save-Replace** command ( ) to replace the previously loaded parameter set with the new parameter set.

   d) You can add a new parameter set as follows: Change the FCU parameters as needed, and use the **Parameter-Save-Insert** command ( ) to insert the new parameter set below the current parameter set.

   e) If the **Continuous operation** option is selected, you can open the Error information panel before starting the operations to monitor the FCU processing. Move the Error information panel so that it does not overlap the FCU main panel.

9. When the desired FX operation is displayed, click **Execute** to start the operation. (If the **Execute** button is not enabled, you have not saved the current parameter set.) If the **Continuous operation** option was selected, FCU will process all operations from the current line to the end of the file and then self-terminate. The error information for these operations is placed in the most recent **.log** file(s) in the current directory (e.g., **mto.log**).

   WARNING: In some early versions of FCU for Windows 2000/2003/Windows NT systems, FCU may overwrite existing Windows 2000/2003/Windows NT systems target files without requesting confirmation.

10. If you started an FXmto operation and the target file already exists, FCU requests overwrite confirmation. Click **OK** to overwrite the target file, or click **Cancel** to cancel the operation.

11. When FCU starts the operation, the Execute panel opens and displays the progress of the operation. To cancel the operation in progress, select **Cancel**.

    Note: The Execute dialog panel will not appear when the mainframe OS is VSE.

12. When the operation is complete, the Execute panel displays the result. If an error occurred, the Error information panel opens automatically to display the error. See Error Codes and Messages for further information on errors.

13. FCU does not load the next operation automatically. To perform another FX operation, select the desired **Parameter-Load** command, and repeat steps (8) through (12). To exit FCU, select the **File-Exit** command.

Note: After an FXotm file transfer from Windows 2000/2003/Windows NT systems, there will be a delay before you can access the FX volume. The length of delay varies according to individual server performance.

Note: After you expand open volumes (LUSE), you will need to reboot Windows 2000/2003/Windows NT systems.

Note: Do not use the open system host to access an FX volume. Use only FAL to access FX volumes. This applies to PC server platforms (e.g., NT) and UNIX-based systems.

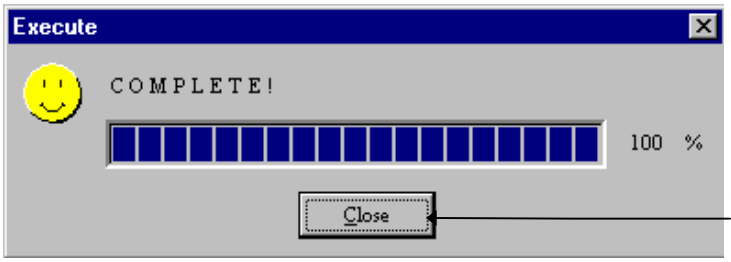


Figure 5-7 Execute Panel Showing Normal End

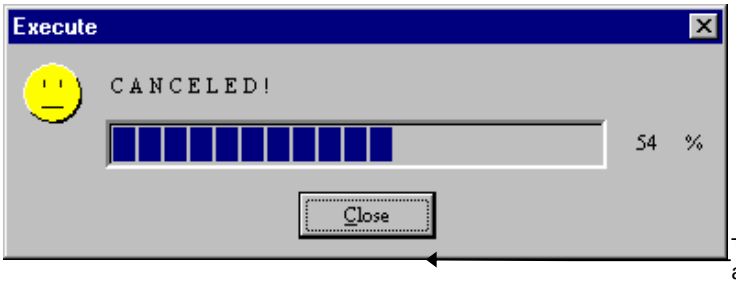


Figure 5-8 Execute Panel Showing Canceled Operation

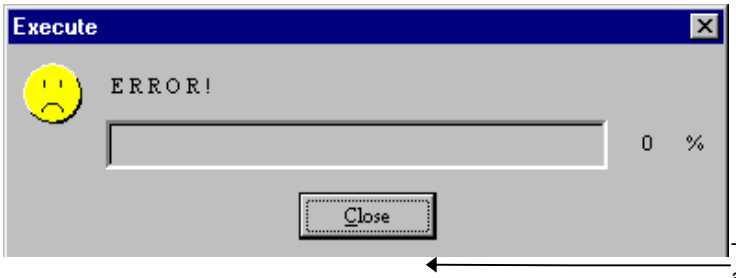


Figure 5-9 Execute Panel Showing Error End

Creating FCU Parameter Definition Files (Windows)

To create an FCU parameter definition file using the FCU GUI for Windows 2000/2003/Windows NT systems:

1. Start the FCU GUI for Windows 2000/2003/Windows NT systems (see section Starting the FCU GUI). If you start FCU from the DOS prompt, enter **fcu** (do not specify the **-nc** or **param** option).
2. When the FCU main panel opens, the title bar should display **Untitled** to indicate that a new parameter definition file is open. If a file name is displayed instead of **Untitled**, use the **File-New** command ( ) to open a new parameter definition file.
3. If you plan to perform FX operations while you are creating the FCU parameter definition file, open the Volume information panel (select **View-Volume information...**), and make sure that the desired FX volume(s) is/are available. If not, FCU will not be able to perform FX operations, but you can still create a new parameter definition file.

4. Enter the desired FCU initiation parameters for the first FX operation.
   - Use the buttons to select the transfer direction (**M to O** or **O to M**).
   - Enter the source and target datasets/files in the **Mainframe File** field (VSN:dataset), and/or **Open-system file** field (/directory/…/filename).
   - Open the Option panel using the **View-Option…** menu command ( ), and then select the desired FCU options (code conversion, padding, delimiters, etc.). Do not select **Continuous execution** or **Clear log file** when creating a new FCU parameter definition file. Close the Option panel when you are done.
5. When the FCU initiation parameters are correct, select the **Parameter-Save-Insert** command ( ) to save the current parameter set in the new FCU parameter definition file. The status bar now displays 1/1 to indicate that line one of one is now being displayed. If the FX volume is available, you can perform the operation now by clicking **Execute**. If the **Execute** button is not enabled, the parameter set has not been saved in the file.
6. Repeat steps (4) and (5) to add each new FCU initiation parameter set to the new FCU parameter definition file. If desired, you can use the **Parameter-Wipe** menu command ( ) to clear the screen before you enter the next set of parameters, or you can leave the parameters and make changes where needed to specify the next new line in the file. Make sure to keep FX operations which use OPEN-*x* FMT volumes in a separate FCU parameter definition file from operations which use -A, -B, and -C volumes.
7. If you need to insert a new line between two existing lines, go to the line above/before the line to be inserted using the **Parameter-Load** commands ( ), change the parameters as needed, and then use the **Parameter-Save-Insert** command ( ) to insert the new line. The new line is inserted below/after the current line number.
8. If you need to modify an existing line, go to the line to be modified using the **Parameter-Load** commands, change the parameters as needed, and then use the **Parameter-Save-Replace** command ( ) to modify the line as specified.
9. When you want to save your new FCU parameter definition file, select the **File-Save** menu command ( ). The file extension must be **.prm**.

Creating Multiple Volume Definition Files (Windows)

Multiple volume Dataset is supported for FXmto, when the version is 01-*XX*-50/YY (*XX*=01 or 02) or later. Multiple volume Dataset is also supported for FXotm when the version is 01-*XX*-60/YY (*XX*=02, 03 or 04) or later. The name of the multiple volume definition file is ***"multidef.dat***". It must be placed under the current directory where the FAL/FCU is to be executed. Specify the other name in "FAL_MULTI_DEF_FILE" of the environment variable, if the name of the multiple volume definition file want to be changed.

Note: Multiple volume Dataset is supported in the case of FileExchangeotm when the version is 01-*XX*-60/YY(*XX*=02, 03 or 04) or higher.

Table 5-2 illustrates the VSN function.

Table 5-2 VSN Function

```
VSN:DSN[,VOLID1] VSN[,VOLID2]  -----  VSN[,VOLIDn]
i)               ii)                  iii)        iv)
end
v)
```

j. i) This parameter is the information of the head volume:

- VSN: a volume serial number with six digit of alphabet (A-Z, @, #, and \) or numeral (0-9) characters.
- DSN: dataset name.(Use maximum 44-digit of alphabet or numeral character)
- VOLID1: a VSN identification (Omit this parameter if a VSN identification is omitted in the volume definition file, and specify same as the volume definition file, if a VSN identification is specified in the volume definition file.)

k. ii) This parameter is the information of the second volume:

- VSN: a volume serial number with six digit of alphabet (A-Z, @, #, and \) or numeral (0-9) characters.
- VOLID2: a VSN identification (Omit this parameter if a VSN identification is omitted in the volume definition file, and specify same as the volume definition file, if a VSN identification is specified in the volume definition file.)

l. iii) This parameter is the information of the last volume.(The number of volume is 'n'):

- VSN: a volume serial number with six digit of alphabet (A-Z, @, #, and \) or numeral (0-9) characters.

- VOLIDn: a VSN identification (Omit this parameter if a VSN identification is omitted in the volume definition file, and specify same as the volume definition file, if a VSN identification is specified in the volume definition file.)

m. iv) Each line above must be separated by using ***"Return"*** key.

n. v) The ***"end"*** specifies that the volume definition file ends here.

Note: Each parameter must be separated with one or more ***"space"*** character(s).

Note: One data set information must be specified in one line.

Note: 999 information can be specified in the multiple volume definition file.

Using FCU from the Command Line (Windows)

To perform FX file transfer operations for Windows 2000/2003/Windows NT systems in a non-GUI environment:

1. Log-in as a user who has administrator privileges.
2. Open the command prompt (DOS Windows) and input the parameters below:
   - **fcunw** [-cl] [param] ([-v])
   - -cl : Specifying that all the log file for FCU will be cleared before stating FCU.
   - param :This parameter is used as same as (1) in this section for UNIX.

   Note: When this parameter is not specified, the file name of the "Parameter definition file" will be assumed to be "fcudata.prm" and it will attempt to read the detail parameters from the file.

   - -v : This displays the version of **fcunw**.

   Note: This parameter cannot be used with any other parameter simultaneously, as it specifies versions.

   - [Return value] 0 : Normal end
   - [Return value] 1 : Error end.

   Note: The **fcunw** command requires the "Parameter definition file" to function properly. If there is no "Parameter definition file" or if there is an incorrect parameter in the "Parameter definition file", the following message will be displayed:

   - [A parameter definition file doesn't exist, or it is illegal.]

6

Performing File Access Library (FAL) Operations

This chapter describes how to perform FAL operations using FX.

- FAL Requirements
- FAL Functions
- Using the FAL Functions
- Multi-Thread Function
- Compiling
- Error Information
- FAL Usage Scenario

The FAL component of FX consists of the object module file **fal.o** (**fal.obj** for Windows 2000/2003/Windows NT systems) and the header file **dataset.h**. The FAL provides several important C functions (Visual C++ for Windows 2000/2003/Windows NT systems) which enable user applications on the open-system hosts to access z/OS data on the TagmaStore USP and NSC FX volumes. There are two types of FAL, the 32bit FAL and the 64bit FAL. The latter is provided by the TagmaStore subsystem.

FAL Requirements

The FAL functions have the same dataset requirements as FCU (e.g., SAM, standard R0 track format). The FAL also has the following additional requirements:

- The FAL functions support only standard MVS™ VTOC. The FAL functions cannot access MVS™ datasets managed by an index VTOC and cannot access VSE datasets when called from user applications.
- The FAL functions are not “thread-safe.” The FAL functions may not operate properly when used by multiple threads within a single process.
- The FAL functions cannot be used by a “signal handler.” If this accidentally happens and memory space is occupied, use **kill** to cancel the processes, and use **ipcrm** to delete the shared memory areas that have KEY=0 (refer to OS manuals). Do not issue the following signals to an FAL/FCU process (UNIX only):

  SIGUSR1, SIGUSR2, SIGILL, SIGTRAP, SIGIOT, SIGABRT, SIGEMT, SIGFPE, SIGKILL, SIGBUS, SIGSEGV, SIGSYS, SIGALRM, SIGPOLL, SIGIO, SIGSTOP, SIGTSTP, SIGCONT, SIGTTIN, SIGTTOU, SIGVTALRM, SIGPROF, SIGXCPU, SIGXFSZ, SIGWAITING, SIGLWP, SIGFREEZE, SIGTHAW, SIGCANCEL
- The following terminology is reserved for the FAL functions and cannot be used in function names, variable names, or constant symbols in the user application:
  - Words that begin with **dataset** or **fast_**
  - **GetVolSers**

FAL Functions

The FAL includes the following C functions (Visual C++ for Windows 2000/2003/Windows NT systems):

- Opening a dataset: **datasetOpen** (see section Opening a Dataset)
- Reading one record from a dataset: **datasetGet** (see section Reading Data)
- Writing one record to a dataset: **datasetPut** (see section Writing Data)
- Closing a dataset: **datasetClose** (see section Closing a Dataset)
- Acquiring error information: **datasetGetLastError** (see section Acquiring Error Information)
- Acquiring dataset attribute information (see section Acquiring Dataset Attributes):
  datasetGetFileInformation **datasetFindNextFile**
  datasetFindFirstFile **datasetFindClose**

Converting Dataset Attribute Information

See section Converting DO and RF Information):

datasetGetDsorgString datasetGetRecfmString

Opening a Dataset

datasetHandle = datasetOpen (pathname, mode)

The **datasetOpen** function opens the dataset specified by **pathname** for the type of access specified by **mode**. Table 6-1 shows the **datasetOpen** arguments and return values.

Table 6-1 DatasetOpen Function

Item	Value	Type	Description
Argument	pathname	char *	VSN:Dataset name VSN = 6-character volser. Volume must be listed in FX volume definition file. Delimiter = : (colon, no spaces allowed) Dataset name: 44 characters max, no spaces allowed.
	mode	char *	r = open dataset for read access w = open dataset for write access
Return value	datasetHandle -1	DATASET_HANDLE	Handle Error end

When the **datasetOpen** function terminates successfully, it returns a handle which identifies the dataset opened. The **datasetHandle** information is used as the argument of subsequent functions such as **datasetGet**, **datasetPut**, or **datasetClose**. Do not change the **datasetHandle** value returned by this function. If the **datasetOpen** function terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function (see section Acquiring Error Information).

The **datasetOpen** function has the following restrictions:

- Only one dataset at a time can be open within one process. When multiple datasets need to be opened, the open dataset must be closed before another dataset can be opened. This restriction does not apply to open-system files.
- A dataset which is being accessed by the **datasetFindFirstFile** or **datasetFindNextFile** function cannot be opened. The **datasetFindClose** function must be executed before the dataset can be opened. This restriction does not apply to open-system files.

Reading Data

reclen = datasetGet (datasetHandle, buf, buflen)

The **datasetGet** function reads one record from the specified dataset (**datasetHandle**) and puts the record into a buffer (**buf**) of length **buflen**. The **datasetGet** function extracts only the data entity from each record and does not transfer the BL and RL bytes for variable-length records to the buffer. Table 6-2 shows the **datasetGet** arguments and return values.

Table 6-2 DatasetGet Function

Item	Value	Type	Description
Argument	datasetHandle buf buflen	DATASET_HANDLE char * long	The datasetHandle value returned by the **datasetOpen** function. Buffer area for storing the read data. Size of the buffer area. If the record is larger than buflen or equal to zero, datasetGet returns an error and does not transfer any data to the buf.
Return value	reclen -1	long	Data entity size transferred to the buffer Error end

Figure 6-1 shows the format requirements for variable-length records accessed by the **datasetGet** function. Each variable-length block must start with the two-byte BL field, and each variable-length record must start with the two-byte RL field. The **datasetGet** function automatically extracts the data entities without the BL and RL fields.

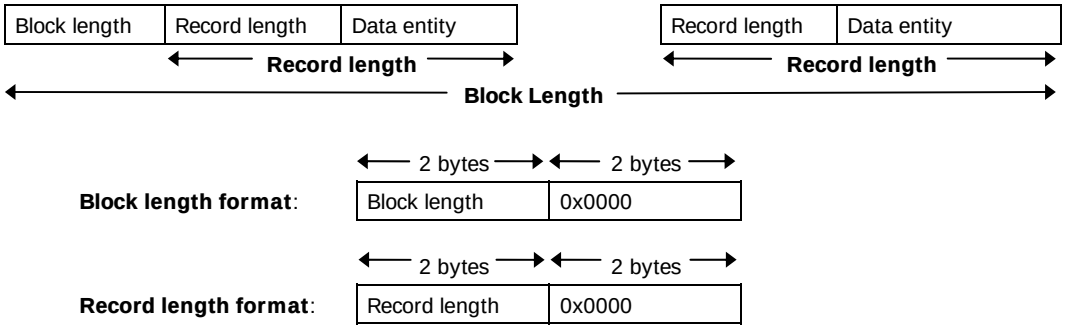


Figure 6-1 Format Requirements for Reading Variable-Length Records

When the **datasetGet** function terminates successfully, it returns the length of the data entity read from the dataset. If the **datasetGet** function detects the end of dataset (EOF) or terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function (see section Acquiring Error Information). For example, when the **datasetGet** function detects EOF, **datasetGetLastError** will return **DATASET_ERROR_END_OF_FILE**.

Writing Data

reclen = datasetPut (datasetHandle, buf, buflen)

The **datasetPut** function writes one record from the **buf** into the dataset specified by **datasetHandle**. Table 6-3 shows the **datasetPut** arguments and return values.

Table 6-3 DatasetPut Function

Item	Value	Type	Description
Argument	datasetHandle buf buflen	DATASET_HANDLE char * long	The datasetHandle value returned by the **datasetOpen** function. Buffer area for storing the write data. Size of the buffer area. If any of the following conditions is detected, datasetPut returns an error and does not transfer any data to the dataset: For fixed-length record: buflen ≠ RL of the dataset For variable-length record: (buflen + 4) > RL of dataset For variable-length record: buflen = 0 (no data entity)
Return value	reclen -1	long	Data entity size written into the dataset. Error end

Figure 6-2 shows the format requirements for variable-length records accessed by the **datasetPut** function. When the target dataset is variable-length, the **datasetPut** function takes the data entity from the **buf**, automatically adds the two-byte RL field, and writes the record into the dataset. When the data is written into the dataset, multiple records are blocked within the extent defined by the VTOC of the dataset.

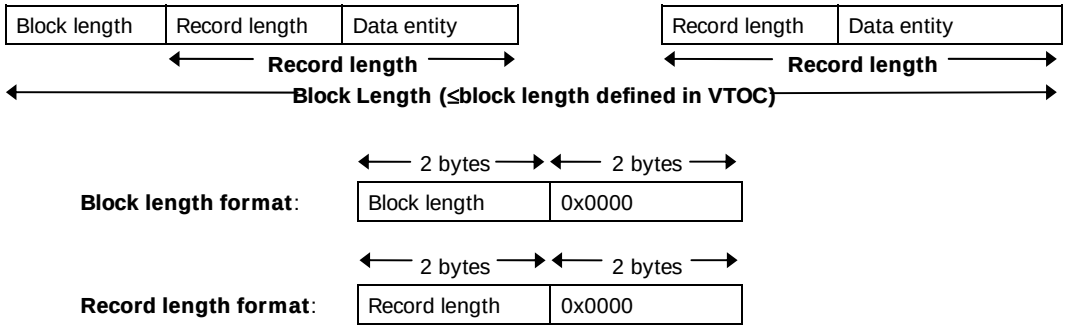


Figure 6-2 Format Requirements for Writing Variable-Length Records

When the **datasetPut** function terminates successfully, it returns the length of the data entity written to the dataset. If the **datasetPut** function terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function (see section Acquiring Error Information).

Closing a Dataset

datasetError = datasetClose (datasetHandle)

The **datasetClose** function closes the dataset specified by **datasetHandle**, which is returned by the **datasetOpen** function. Table 6-4 shows the **datasetClose** arguments and return values.

Table 6-4 DatasetClose Function

Item	Value	Type	Description
Argument	datasetHandle	DATASET_HANDLE	The datasetHandle value returned by the **datasetOpen** function.
Return value	0 -1		Normal end Error end

When the **datasetClose** function terminates successfully, it returns a value of 0. If it terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function (see section Acquiring Error Information).

Acquiring Error Information

datasetLastError = datasetGetLastError()

The **datasetGetLastError** function acquires the error code information for the most recent error. Errors in FAL functions are defined in **dataset.h** (see Appendix C). Errors in UNIX are defined by a standard error file (**errno.h**). Errors in Windows 2000/2003/Windows NT systems are defined by **errno.h** attached with Microsoft Visual C++. Table 6-5 shows the **datasetClose** arguments and return values.

Table 6-5 DatasetGetLastError Function

Item	Value	Type	Description
Argument	none	—	—
Return value	datasetLastError	Long	Error code

Acquiring Dataset Attributes

FAL provides several functions for acquiring the complete dataset attribute information for one or more datasets: **datasetGetFileInformation**, **datasetFindFirstFile**, **datasetFindNextFile**, and **datasetFindClose**. The dataset attribute information returned by these functions contains:

```
typedef struct DATASET_FIND_DATA {
   unsigned short blockSize;    /* Block length       */
   unsigned short recordSize;   /* Record length      */
   unsigned char dsorg[2];      /* dataset type       */
   unsigned char recfm;  /* record format      */
   char name[44];/* dataset name       */
   unsigned short lastBlockTt;  /* last block address (relative track number) */
   unsigned char lastBlockR;    /* last block address (relative record number)*/
}  DATASET_FIND_DATA;
```

Acquiring Attribute Information for a Specific Dataset

datasetError = datasetGetFileInformation (pathname, &ffd)

The **datasetGetFileInformation** function acquires the attribute information for the dataset specified by **pathname** and returns the data into **ffd**. Table 6-6 shows the **datasetGetFileInformation** arguments and return values.

Table 6-6 DatasetGetFileInformation Function

Item	Value	Type	Description
Argument	pathname ffd	char * DATASET_FIND_DATA	VSN:Dataset name VSN = 6-character volser. Volume must be listed in volume definition file. Delimiter = : (colon, no spaces) Dataset name: 44 characters max, no spaces. Area where the dataset attribute information is stored.
Return value	0 -1		Normal end Error end

When the **DatasetGetFileInformation** function terminates successfully, it returns a value of 0. If it terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function (see section Acquiring Error Information).

The **DatasetGetFileInformation** function has the following restriction:

- The **DatasetGetFileInformation** function cannot be used on an open dataset. Use this function before opening or after closing the dataset.

Acquiring Attribute Information for Multiple Datasets

A combination of the **datasetFindFirstFile**, **datasetFindNextFile**, and **datasetFindClose** functions is used to acquire attribute information for more than one dataset in the specified z/OS volume.

1. **datasetHandle = datasetFindFirstFile (pathname, &ffd)**

   The **datasetFindFirstFile** function acquires the attribute information for the first dataset in the volume specified by **pathname** and returns the data into **ffd**. Table 6-7 shows the **datasetFindFirstFile** arguments and return values.

Table 6-7 DatasetFindFirstFile Function

Item	Value	Type	Description
Argument	pathname ffd	char * DATASET_FIND_DATA	VSN VSN = 6-character volser. Volume must be listed in volume definition file. Area where the dataset attribute information is stored.
Return value	datasetHandle -1	DATASET_HANDLE	Normal end Error end

When the **datasetFindFirstFile** function terminates successfully, it returns a handle which identifies the dataset for which the attribute information was acquired. The **datasetHandle** information is used as the argument of the subsequent functions **datasetFindNextFile** and **datasetFindClose**. Do not change the **datasetHandle** value returned by this function. If the **datasetFindFirstFile** function terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function (see section Acquiring Error Information). For example, when the **datasetFindFirstFile** function does not find any datasets in the VTOC, the **datasetGetLastError** function will return **DATASET_ERROR_END_NO_DATASET**.

The **datasetFindFirstFile** function has the following restrictions:

– The **datasetFindFirstFile** function cannot be used on an open dataset. Use this function before opening or after closing the dataset.

– After a dataset has been accessed by the **datasetFindFirstFile** function, the dataset cannot be opened until after the **datasetFindClose** function is called.

2. datasetError = datasetFindNextFile (datasetHandle, &ffd)

The **datasetFindNextFile** function acquires the attribute information for the next dataset in the volume specified by **datasetHandle** and returns the data into **ffd**. This function can be repeated until "no dataset found" is returned, or until the user application determines that no more information is needed. Table 6-8 shows the **datasetFindNextFile** arguments and return values.

Table 6-8 DatasetFindNextFile Function

Item	Value	Type	Description
Argument	datasetHandle	DATASET_HANDLE	The datasetHandle value returned by the preceding **datasetFindFirstFile** function.
	ffd	DATASET_FIND_DATA	Area where the dataset attribute information is stored.
Return value	0		Normal end
	-1		Error end, or no dataset found

When the **datasetFindNextFile** function terminates successfully, it returns a value of 0. If this function terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function (see section Acquiring Error Information). For example, when the **datasetFindNextFile** function cannot find the next dataset in the VTOC, the **datasetGetLastError** function will return **DATASET_ERROR_END_OF_VTOC**.

The **datasetFindNextFile** function has the following restrictions:

- The **datasetFindNextFile** function cannot be used on an open dataset. Use this function before opening or after closing the dataset.
- After a dataset has been accessed by the **datasetFindNextFile** function, the dataset cannot be opened until after the **datasetFindClose** function is called.
- The **datasetFindFirstFile** function must be called prior to **datasetFindNextFile**.

3. datasetError = datasetFindClose (datasetHandle)

The **datasetFindClose** function terminates the acquisition of attribute information by the **datasetFindFirstFile** and **datasetFindNextFile** functions and closes the dataset. The **datasetFindFirstFile** function must be called prior to **datasetFindClose**. Table 6-9 shows the **datasetFindClose** arguments and return values.

Table 6-9 DatasetFindClose Function

Item	Value	Type	Description
Argument	datasetHandle	DATASET_HANDLE	The datasetHandle value returned by the preceding **datasetFindFirstFile** function.
Return value	0		Normal end

	-1		Error end

When the **datasetFindClose** function terminates successfully, it returns a value of 0. If this function terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function (see section Acquiring Error Information).

Converting DO and RF Information

The FAL provides two functions for converting specific attribute information from a dataset into character strings: **datasetGetDsorgString**, and **datasetGetRecfmString**.

Converting the Dataset Organization (DO) Type Value

datasetError = datasetGetDsorgString (dsorg, text)

The **datasetGetDsorgString** function converts the dataset organization (DO) type to a three-byte character string. The DO type is specified by **dsorg[2]** in **DATASET_FIND_DATA**. Table 6-10 lists the **datasetGetDsorgString** arguments and return values.

Table 6-10 DatasetGetDsorgString Function

Item	Value	Type	Description
Argument	dsorg text	u_char[ ] char [3]	Value of dsorg[2] (two bytes) obtained by the attribute acquisition function **datasetGetFileInformation**, **datasetFindFirstFile**, or **datasetFindNextFile**. Character string indicating the dataset organization (DO) type (3 bytes): PS physical sequential organization VS VSAM organization DA direct access organization PO Partial organization ** Other than above types
Return value	0 -1		Normal end Error end

Converting the Record Format (RF) Type Value

datasetError = datasetGetRecfmString (recfm, text)

The **datasetGetRecfmString** function converts the record format (RF) type to a five-byte character string. The RF type is specified by **recfm** in **DATASET_FIND_DATA**. Table 6-11 lists the **datasetGetRecfmString** arguments and return values.

Table 6-11 DatasetGetRecfmString Function

Item	Value	Type	Description

Argument	recfm text	u_char char [5]	Value of recfm (one byte) obtained by attribute acquisition function **datasetGetFileInformation**, **datasetFindFirstFile**, or **datasetFindNextFile**. Character string (5 bytes) indicating the (RF) type: text[0] F fixed-length record V variable-length record U unknown-length record text[1] B blocking record sp spanned record st standard format record
Return value	0 -1		Normal end Error end

Using the FAL Functions

The FAL functions can be executed by any C program on the UNIX host. The FAL does not support C++®. The z/OS datasets accessed by the FAL functions must be located on TagmaStore USP or NSC FX volumes. The FX volumes must be installed and configured correctly (see section Installing and Configuring the FX Volumes), the FAL/FCU software must be installed properly (see section Installing the FAL/FCU Software), and the FX volume definition file must be available and configured correctly. Since FAL operations do not involve GUI windows, the X windows environment and FcuMf resource file are not required.

Figure 6-3 shows an example of reading data using the FAL functions. Figure 6-4 shows an example of acquiring attribute information using the FAL functions. To use the FAL functions in a C program (Visual C++ for Windows 2000/2003/Windows NT systems):

1. Copy the FX volume definition file (**datasetmount.dat**) to the directory containing the C program that will call the FAL C function(s).
2. Include the FAL header file (**dataset.h**) within the C program that will call the FAL function(s) (e.g., copy **dataset.h** to **/usr/include**).
3. Using 32bit FAL, compile the C program as follows:

   IBM AIX

   # cc -qlanglvl=ansi —D_NO_MT -o Output file name Source file name /usr/lib/libfal.a
   libfal.a = file name of FAL object module

   HP-UX

   # cc -Ae +DAportable —D_NO_MT -o Output file name Source file name /usr/lib/libfal.sl
   libfal.sl = file name of FAL object module

   Solaris

   # cc —D_NO_MT -o Output file name Source file name /usr/lib/libfal.so.1
   libfal.so.1 = file name of FAL object module

   DIGITAL UNIX/Tru64

   # c89 —D_NO_MT -o Output file name Source file name /usr/lib/libfal.so
   libfal.so = file name of FAL object module

   Linux

   # gcc —D_NO_MT -o Output file name Source file name /usr/lib/libfal.so.1
   libfal.so.1 = file name of FAL object module

4. Using 64bit FAL, compile the C program as follows:

   IBM AIX

 # cc -qlanglvl=ansi -q64 —D_NO_MT -o Output file name Source file name /usr/lib/libfal64.a

libfal64.a = file name of FAL object module

HP-UX

 # cc -Ae +DAZ.0W —D_NO_MT -o Output file name Source file name /usr/lib/pa20_64/libfal64.sl

libfal64.sl: = file name of FAL object module.

Solaris

 # cc xarch=v9 —D_NO_MT -o Output file name Source file name
/usr/lib/sparcv9/libfal64.so.1

libfal64.so.1: = file name of FAL object module.

Linux

 # gcc -o Output file name Source file name
/usr/lib/libfal.so.1

libfal.so.1:

This specifies a file name of the object module of the File Access Library.

Windows 2000/2003/Windows NT systems (Visual C++)

Start **Developer Studio** and create a new project.

a) Copy the following three FAL files into the project folder: **dataset.h**, **fal.dll**, **fal.lib**

b) Select **Settings** in the Projects menu of Developer Studio.

c) On the Project Settings panel, select the **Link** tab.

d) Enter **fal.lib** in the object/library module field.

e) Select the **C/C++** tab in Project settings dialog.

f) Add **_NO_MT** to the **preprocessor definitions** field.

e) Build and execute.

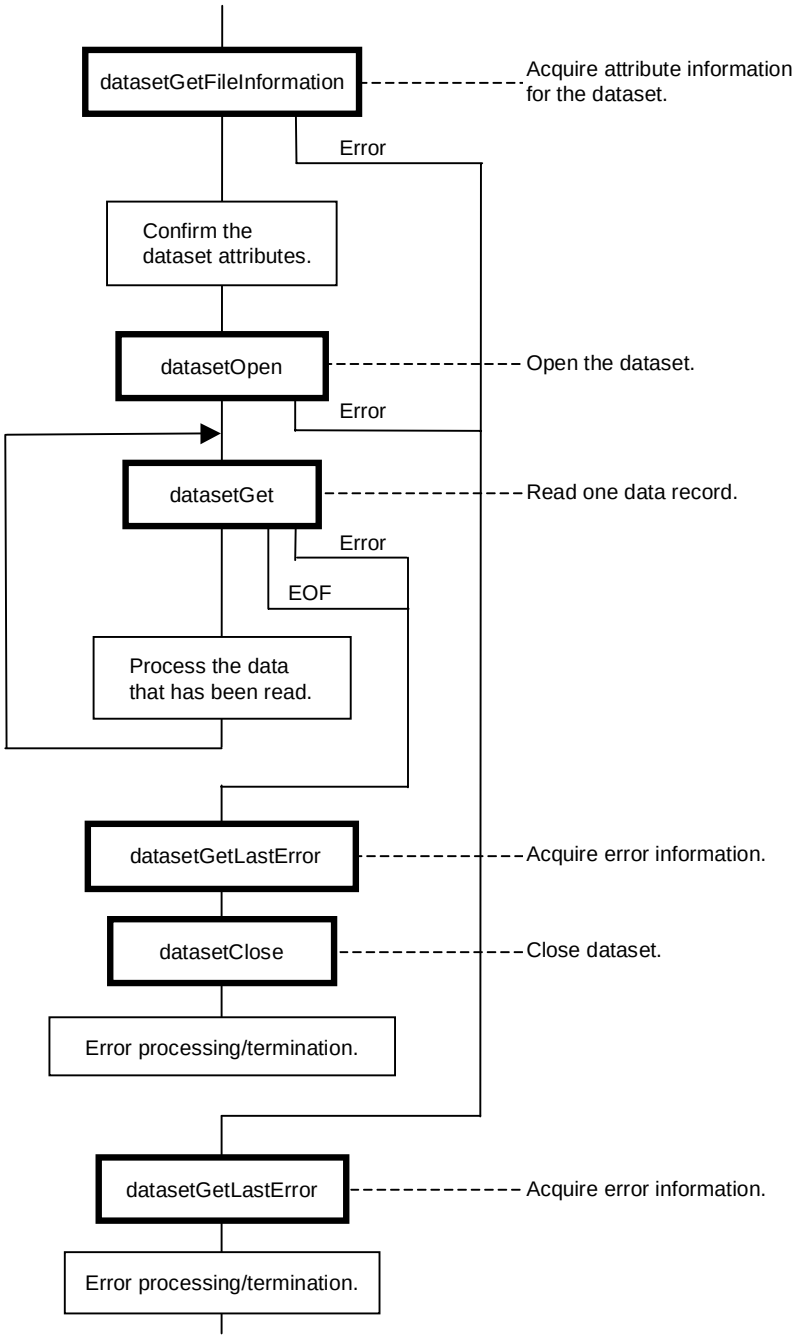


Figure 6-3 Example of Reading Data from an z/OS Dataset Using FAL

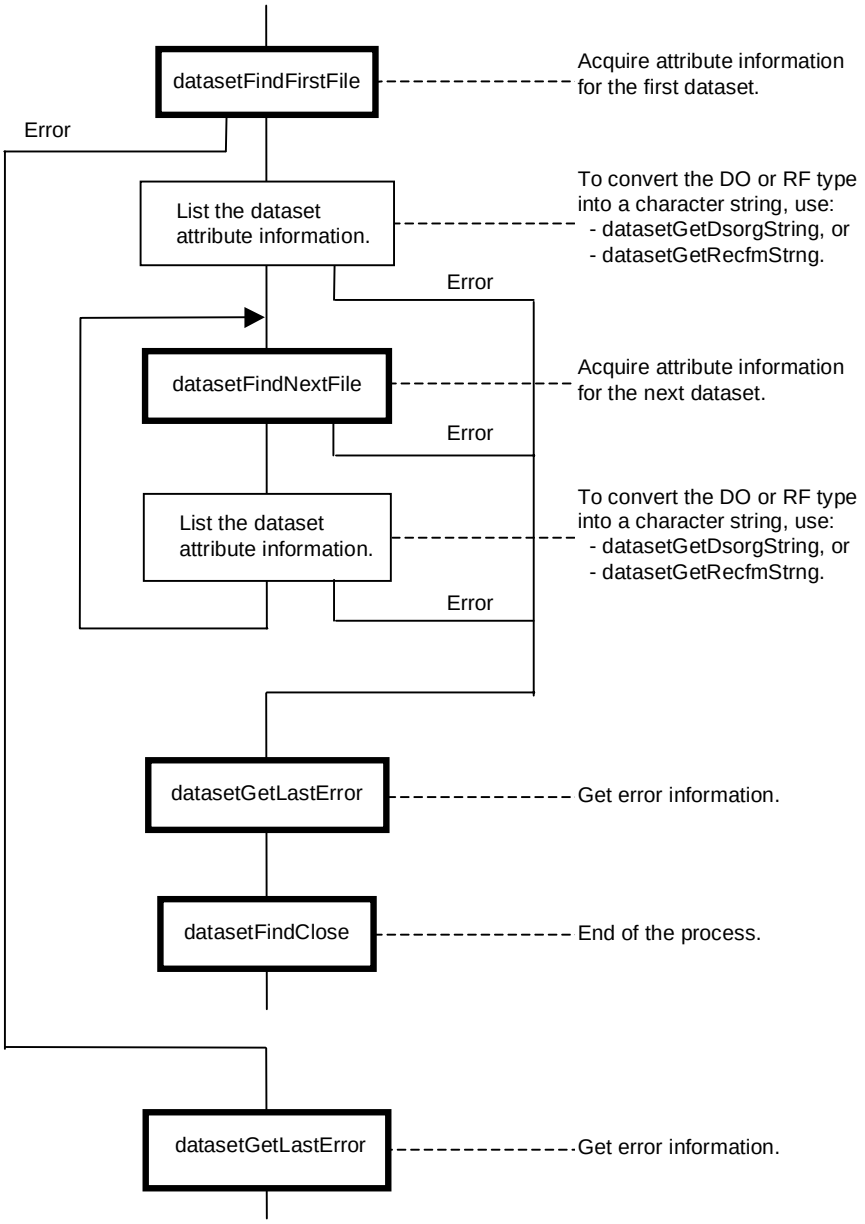


Figure 6-4 Example of Acquiring z/OS Dataset Attributes Using FAL

Multi-Thread Function

Specifications: FAL provides the following functions (C programming language):

- Information storage area: **dataset_AllocGlobal**
- Open a dataset specified by volume name and dataset name: **dataset_Open**
- Read a record specified by dataset: **dataset_Get/dataset_Get2**
- Write a record specified by dataset: **dataset_Put/dataset_Put2**
- Close a specified dataset: **dataset_Close**
- Free storage area: **dataset_FreeGlobale**
- Return a file pointer to top: **dataset_Rewind**
- Get a dataset attribute: **dataset_GetFileInformation**, **dataset_FindFirstFile**, **dataset_FindNextFile**, **dataset_FindClose**

Programming Restrictions:

- You cannot use FX from the Signal Handler.
- The words listed below are reserved words. When the user creates a program using FAL, these words cannot be used for function names, variable names, symbol names, or constant names:
  - dataset
  - fast_
  - GetVolSers
- Do not mix the FX multi-thread function with user API for multi-thread and user API for non-multi-thread.
- This function is only applicable for AIX(32bit version) and Windows NT/Windows 2000/Windows 2003.
- You do not need a volume definition file when user uses API for multi-thread.
- You can open multiple datasets simultaneously using multi-thread API:
  - **data set_AllocGlobal**: reserve an area for information of dataset “A”.
  - **dataset_AllocGlobal** : reserve an area for information of dataset “B”.
  - **datset_Open**: Open dataset “A”.
  - **dataset_Open**: Open dataset “B”.

Information Storage Area

Format: **memError= dataset_AllocGlobal(dgpp,derrno,malloc,free)** (Table 6-12)

Table 6-12 Arguments, Types and Descriptions for Information Storage Area

Argument	Type	Description
dgpp	void (see ***Note 2***)	Global memory area
derrno	long (see ***Note 1***)	An error information stored area
malloc	void (see ***Note 2***)	malloc()
free	void (see ***Note 1***)	free()
Return value: memError 0	int	Abnormal end

Note 1: When you issue this function, you must issue **dataset_FreeGlobal()** in the end process.

Note 2: You must issue this function before **dataset_Open()** and **dataset_FindFirstFile()**.

- Arguments:
  - **dgpp**: Global memory area stored area
  - **derrno**: Return an address stored FAL error code
  - **malloc**: Specify an address of malloc function. Specify as malloc.
  - **free**: Specify an address of free function. Specify as free.
- Return Value:
  - When this function ends normally, it returns a **1**.
  - When this function ends abnormally, it returns a **0**. For further information, refer to Appendix B.
- ***Example:***

```
void *memptr;  /* global memory area */
long err;        /* global err information */
int retcode;
     :
retcode = dataset_AllocGlobal(&memptr, &err, malloc, free);
     :
retcode = dataset_FreeGlobal(&memptr, &err)
```

Open Dataset

Format:
datasetError=dataset_Open(global,g_error,devname,dsname,voltype, mode)

Table 6-13 Arguments, Types and Descriptions for Open Data Set

Argument	Type	Description
global	void (see ***Note 1***)	Global memory area
g_error	long (see ***Note 1***)	An error information stored area
devname	char (see ***Note 1***)	raw device name
dsname	char (see ***Note 1***)	dataset name
voltypr	char (see ***Note 1***)	volume emulation type
mode	char (see ***Note 1***)	open mod
Return value: datasetError	long	Abnormal end
-1		

Note 1: When you issue this function, you must issue **dataset_ Close()** in the end process.

Note 2: You must issue this function before **dataset_Open()**, **Get()**, **dataset_Get2()**, **dataset_Put()**, **dataset_Put2()**, **dataset_Rewind(),** and **dataset_GetFileInformation()**.

This function opens a specified dataset (file) with a specified open mode.

- Argument:
  - **global**: global memory area (specify a Global memory area gotten by **dataset_AllocGlobal**)
  - **g_error**: Specify an address to store FAL error code.
  - **devname**: raw device name (special file)
  - **dsname**: dataset name
  - **voltype**: Device emulation type (3390-3A/9A/LA,and 3390-3B/9B/LB, 3380-3A, 3380-3B)
  - **mode**:
    - "r": Read only
    - "w": Write only
  - **Return Value**: When this function ends abnormally, it returns -1.
- ***Example:***

```
void *memptr;                   /* global memory area */
 long err,datasetError;         /* global err information */
```

```
    int retcode;
    retcode = dataset_AllocGlobal(&memptr, &err, malloc, free);
     :
    datasetError = dataset_Open(memptr, &err,"
 HYPERLINK "\\\\.\\PHYSICALDRIVE1"
\\.\PHYSICALDRIVE1
,"DSN001
, "3390-3A","r");
     :
    datasetError=dataset_Close(memptr, &err);
    retcode = dataset_FreeGlobal(&memptr, &err);
```

Read Data

- Format: **reclen = dataset_Get(global, g_error, buf, buflen)**

  reclen= dataset_Get2(global, g_error, buf, buflen)

Table 6-14 Arguments, Types and Descriptions for Read Data

Argument	Type	Description
global	void*	Global memory area
g_error	long*	An error information stored area
buf	char*	Read buffer
buflen	long*	Data length transferred to buffer
Return value: reclen	long*	Data length read to buffer
-1		Abnormal end

Note: This function provides the ability to read a record of a previously opened dataset out to a buffer. Transferred data is real data only. For further information, see section Reading Data.

- Argument:
  - **global:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.)
  - **g_error:** Specify an address to store FAL error code.
  - **buf:** Specify a buffer to store read data.
  - **buflen:** Specify buffer size.
- Return Value:
  - For **dataset_Get():**
    - When this function ends normally, **reclen** (record length) is returned. (1 record length≦32760).
    - When this function ends abnormally, "**- 1**" is returned.
    - When this function detects EOF, "**0**" is returned.
  - For **dataset_Get2():**
    - When this function ends normally, **reclen** (record length) is returned. (0≦record length≦32760).
    - When this function ends abnormally, "**- 1**" is returned.
    - When this function detects EOF, "**DATASET_ERROR_END_OF_FILE** " is returned.

  Note: When "**- 1**" is returned, refer to the content of **g_error** for error code details. For further information, see Appendix B.

Write Data

- Format: **recren= dataset_ Put(global, g_error, buf, buflen)**

  recren= dataset_ Put2(global, g_error, buf, buflen)

Table 6-15 Arguments, Types and Descriptions for Write Data

Argument	Type	Description
global	void (see ***Note 1***)	Global memory area
g_error	long (see ***Note 1***)	An error information stored area
buf	char (see ***Note 1***)	Read buffer
buflen	long (see ***Note 1***)	Data length transferred to buffer
Return value: reclen	long (see ***Note 1***)	Data length read to buffer
-1		Abnormal end

Note 1: When buflen is "**0**", the **dataset_Put** function has ended abnormally, but the **dataset_Put2** function ends normally. (It is possible to handle **0** data.)

Note 2: When a full data error occurs, a return value of **dataset_Put2** is "**- 1**", but a return value of **dataset_Put** is "**Y**". **G_error** is "**- 50**".

This function writes a record of the previous opened dataset to a buffer. For variable length record formats, this function writes real data to a buffer with record length. For more detail, see section Reading Data.

- Argument:
  - **global:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.)
  - **g_error:** Specify an address to store FAL error code.
  - **buf:** Specify a buffer to store write data.
  - **buflen:** Specify buffer size.
- Return Value: When this function ends normally, **reclen** (record length) is returned. When this function ends abnormally, "**- 1**" is returned. When "**- 1**" is returned, refer to the contents of **g_error** for error code details. For further information, see Appendix B.

Close Dataset

- Format: **datasetError=dataset_Close(global,g_error)**

Table 6-16 Arguments, Types and Descriptions for Close Dataset

Argument	Type	Description
global	void (see ***Note 1***)	Global memory area
g_error	long (see ***Note 1***)	An error information stored area
Return value: datasetError -1	long (see ***Note 1***)	Abnormal end

This function closes a dataset.

- Return value:
  - When this function ends normally, "**0**" is returned.
  - When "**- 1**" is returned, refer to the content of **g_error** for error code detail. For further information, see Appendix B.

Free Information Stored Area

- Format: **memError= dataset_FreeGlobal(dgpp, derrno)**

Table 6-17 Arguments, Types and Descriptions for Free Information Stored Area

Argument	Type	Description
dgpp	void*	Global memory area
derrno	long*	An error information stored area
Return value: memError 0	int	Abnormal end

This function releases information stored area.

- Argument:
  - **dgpp:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.)
  - **derrno:** Specify an address to store FAL error code.

- Return Value:
  - When this function ends normally, "**1**" is returned.
  - When this function ends abnormally, "**0**" is returned. When "**0**" is returned, refer to the content of **derrno** for error code detail. For further information, see Appendix B.

Initialize Target Record Pointer

- Format: **datasetError=dataset_Rewind(global,g_error)**

Table 6-18 Arguments, Types and Descriptions for Initialize Target Record Pointer

Argument	Type	Description
global	void (see *Note 2*)	Global memory area
g_error	long (see *Note 1*)	An error information stored area
Return value: datasetError	long	Abnormal end
-1		

Note 1: When this function is issued before **dataset_Put**, **dataset_Put2**, **dataset_Get,** and **dataset_Get2**, the pointer is returned to the top record. And then next **dataset_Put**, **dataset_Put2**, **dataset_Get,** and **dataset_Get2** are performed from the top record.

- Argument:
  - **global:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.).
  - g_**error:** Specify an address to store FAL error code.
- Return Value:
  - When this function ends normally, "**0**" is returned.
  - When this function ends abnormally, "**- 1**" is returned.
  - When "**- 1**" is returned, refer to the contents of **g_error** for error code detail. For further information, see Appendix B.

Get Dataset Attribute Information

Get Specified Dataset Attribute Information

- Format: **datasetError= dataset_GetFileInformation(global, g_error, &ffd)**

Table 6-19 Arguments, Types and Descriptions for Get Specified Dataset Attribute Information

Argument	Type	Description
global	void (see ***Note 2***)	Global memory area
g_error	long (see ***Note 1***)	An error information stored area
ffd	DATASET_FIND_DATA	A dataset attribute information stored area
Return value: datasetError -1	long	Abnormal end

An attribute of the opened dataset is returned to **ffd**.

- Argument:
  - **global:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.).
  - **g_error:** Specify an address to store FAL error code.
  - **ffd:** A first dataset attribute information stored area.

  The dataset attribute information format is shown below:

```
typedef struct DATASET_FIND_DATA {
   unsigned short blockSize;           /* Block length    */
   unsigned short recordSize;          /* Record length   */
   unsigned char dsorg[2];                    /* Dataset type    */
   unsigned char recfm;                       /* Record format   */
   char name[44];                             /* Dataset name    */
   unsigned short lastBlockTt;  /* Last block address(relative track number) */
   unsigned char lastBlockR;    /* Last block address(relative block number) */
      unsigned char mftype;    /* Mainframe OS(MVS·VOS3·MSP/VSE/VOS1/XSP) */
}  DATASET_FIND_DATA;
```

- Return Value:
  - When this function ends normally, "**0**" is returned.
  - When this function ends abnormally, "**- 1**" is returned.
  - When "**- 1**" is returned, refer to the contents of **g_error** for details. For further information, see Appendix B.

  Note: * You must issue **dataset_Open()** before this function.

Get Multiple Dataset Attribute Information

- Format (1): datasetHandle=dataset_FindFirstFile(global, g_error, pathname, voltype, and ffd)

Table 6-20 Arguments, Types and Descriptions for Get Specified Dataset Attribute Information

Argument	Type	Description
global	void**	Global memory area
g_error	long*	An error information stored area
ffd	DATASET_FIND_DATA	A dataset attribute information stored area
Return value: datasetError	long	Abnormal end
-1		

This function returns top dataset attribute information specified by raw device name to ffd. This function is used with **dataset_FindFirstFile**, **dataset_FindNextFile** and **dataset_FindClose**.

- Argument (1):
  - **global:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.).
  - **g_error:** Specify an address to store FAL error code.
  - **Pathname:** Address of partition name/physical drive name.
  - **Voltype:** Device emulation type (3390-3A/9A/LA, and 3390-3B/9B/LB, 3380-3A, 3380-3B).
  - **ffd:** A first dataset attribute information stored area.
- Return Value (1):
  - When this function ends normally, "**DATASET_HANDLE**" is returned. This handler is used as an argument for next **dataset_FindNextFile** and **dataset_FindClose** functions.
  - When this function ends abnormally, "**- 1**" is returned.
  - When "**- 1**" is returned, refer to the contents of **g_error** for error code detail. For further information, see Appendix B.

    Note: When there is no dataset in the VTOC, the **g_error** is "**DATASET_ERROR_NO_DATASET**".
- Format (2): datasetError= dataset_FindNextFile(global,g_error,datasetHandle, &ffd)

Table 6-21 Arguments, Types and Descriptions for Get Multiple Dataset Attribute Information (2)

Argument	Type	Description
global	void (see ***Note 1***)	Global memory area
g_error	long (see ***Note 1***)	An error information stored area
datasetHandle ffd	DATSET_HANDLE	Dataset handler
Return value: datasetError	DATASET_FIND_DATA	A dataset attribute information stored area
-1	long	Abnormal end

This function gets a second dataset and more attribute information. You can get just the next set of dataset attribute information, or you can use this function until no further dataset information is available or returned.

- Argument (2):
  - **global:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.).
  - **g_error:** Specify an address to store FAL error code.
  - **datasetHandle:** Specify dataset handler.
  - **ffd:** Next dataset attribute information stored area.

  Refer to **dataset_GetFileInformation** for dataset attribute information.
- Return Value (2):
  - When this function ends normally, "**0**" is returned.
  - When this function ends abnormally, "**- 1**" is returned.
  - When "**- 1**" is returned, refer to the contents of **g_error** and for error code details. For further information, see Appendix B.
  - When there is no dataset in the VTOC, the return value is "**- 1**" and error information is **DATASET_ERROR_END_OF_VTOC**.

  Note*: You must issue **dataset_FindFirstFile** before this function. When you finish getting dataset attribute information, you must issue **dataset_FindClose** in the end process.
- Format (3): datasetError= dataset_FindClose(global,g_error,datasetHandle).

Table 6-22 Arguments, Types and Descriptions for Get Multiple Dataset Attribute Information (3)

Argument	Type	Description

global	void (see ***Note 1***)	Global memory area
g_error	long (see ***Note 1***)	An error information stored area
datasetHandle	DATSET_HANDLE	Dataset handler
Return value: datasetError -1	long	Abnormal end

This function declares the end of the process, and gets dataset attribute information using **dataset_FindFirstFile** and **dataset_FindNextFile**.

- Argument (3):
  - **global:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.).
  - **g_error:** Specify an address to store FAL error code.
  - **datasetHandle:** Specify dataset handler.
- Return Value (3):
  - When this function ends normally, "**0**" is returned.
  - When this function ends abnormally, "**- 1**" is returned.
  - When "**- 1**" is returned, refer to the content of **g_error** and for error code details. For further information, see Appendix B.

Compiling

An example of installation including FAL is shown below. For UNIX operating systems, you need to use a C language compiler based on ANSI. You need to include a header file in the program which will be using FAL.

- For Windows NT/Windows 2000/2003 systems:
  o. 1. Launch Developer Studio.
  p. 2. Create a new project.
  q. 3. Copy the following FAL files to the project folder/directory:
    - dataset.h
    - falmt.dll
    - falmt.lib
  r. 4. Select **SETTING** on the Developer Studio **PROJECT** menu.
  s. 5. Select the **LINK** tab in the **Project** setting dialog.
  t. 6. Add **falmt.lib** to the **OBJECT/LIBRARY MODULE** column.
  u. 7. Build/Execute.

Note: For AIX systems:

```
#cc -qlanglvl=ansi -o output file name source file name /usr/lib/libfalmt.a
```

libfalmt.a: object module file name of Multi-thread for FAL.

Error Information

For details on error messages, see Appendix B. The following error codes do not occur for FAL Multi-thread:

-2, -6, -20, -23, -32

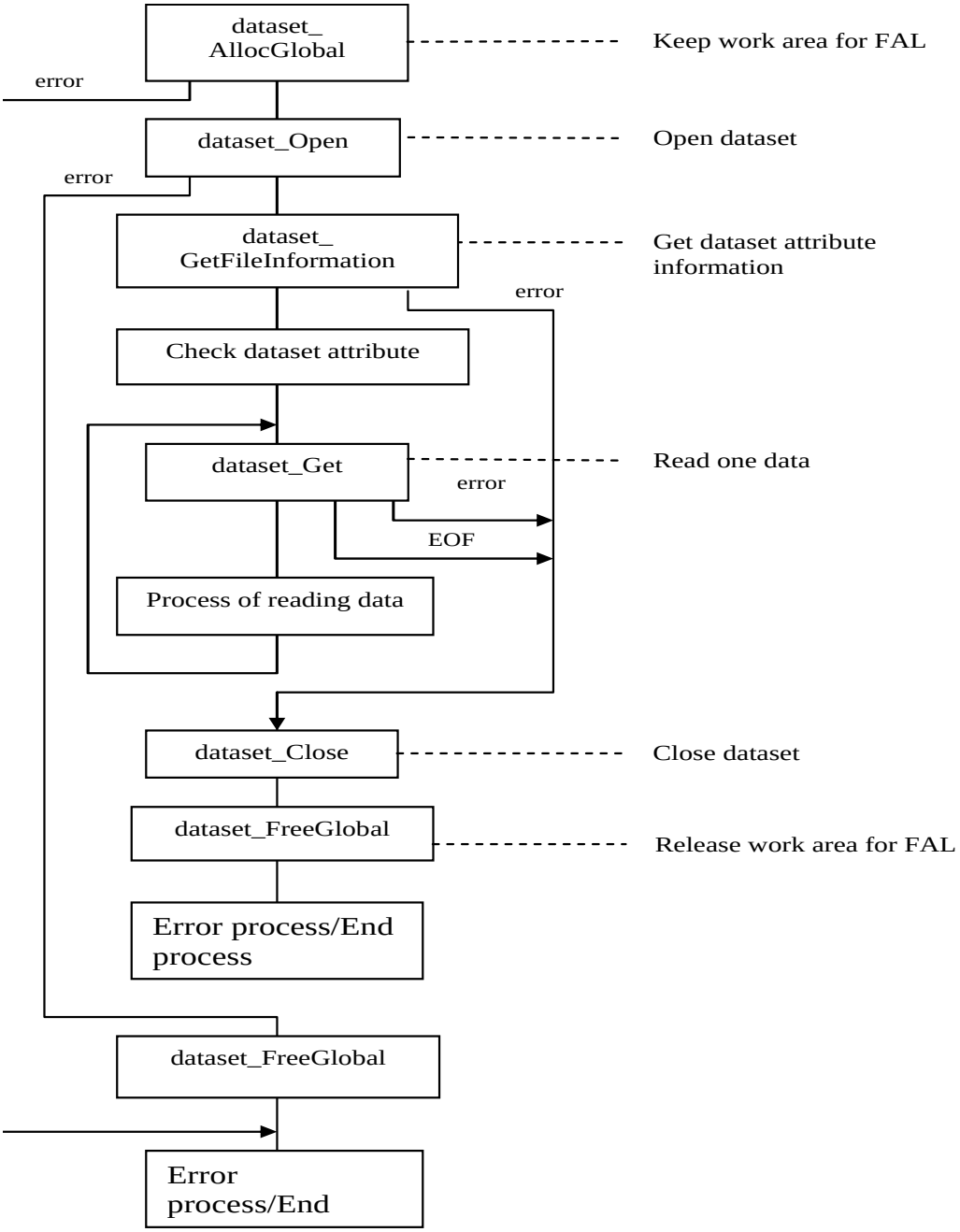
The following error codes **only** occur for FAL Multi-thread:

Table 6-23 FAL Multi-thread Error Codes

-29	DATASET_ERROR_CANNOT_MALLOC malloc() function is abnormally ended.
-30	DATASET_ERROR_FREE_INVALID_AREA Invalid area for global area.
-31	DATASET_ERROR_CANNOT_FREE free() function is abnormally ended.

FAL Usage Scenario

- ***Example 1:*** **read data** flowchart



11

Troubleshooting

This chapter provides descriptions of error codes and support desk contact data.

- ☐ Troubleshooting
- ☐ Error Codes and Messages
- ☐ Calling the Support Center

Troubleshooting

For general troubleshooting information on the TagmaStore USP and NSC subsystems, please refer to the *Hitachi TagmaStore Universal Storage Platform User and Reference Guide* (MK-94RD231) or *Hitachi TagmaStore Network Storage Controller User and Reference Guide* (MK-95RD279).

The FAL/FCU software is not expected to fail in any way. When errors are detected, error codes and messages are displayed and/or logged. Appendix C lists the FAL and FCU error codes and provides instructions for resolving the error conditions.

If you have a problem with the FAL/FCU software, first make sure that the problem is not being caused by other open-system software or hardware, and try rebooting the open-system server.

For FCU operations, make sure that the FX volume definition file and FCU initiation parameters are correct. Table 7-1 lists potential error conditions in FAL/FCU and provides instructions for resolving each condition.

If you are still unable to resolve an error condition, please ask your Hitachi Data Systems representative for help, or call the Hitachi Data Systems Support Center for assistance (see section Error Codes and Messages).

Table 7-1 Troubleshooting

Error Condition	Recommended Action
UNIX files in non- TagmaStore USP and NSC subsystem could not be accessed.	Make sure that the devices have been mounted. If mounting is done during an FCU operation, the results cannot be guaranteed because error information may not be reported to FCU.
Solaris system reports an error indicating **libXm.so.xx is not found**.	Define a path to the Xmlibrary as follows: 1. For C shell, add the following line to the .cshrc file in the home directory: setenv LD_LIBRARY_PATH /usr/dt/lib:$LD_LIBRARY_PATH 2. For non-C shell, add the following two lines to the .dtprofile file in the home directory: LD_LIBRARY_PATH=/usr/dt/lib:$LD_LIBRARY_PATH export LD_LIBRARY_PATH
Windows 2000/2003/Windows NT systems only: FCU reports errors when accessing an FCU parameter definition file.	Remove all space lines from the FCU parameter definition files. FCU versions 01-01-24 and earlier supported space lines when run without the GUI. FCU versions 01-01-36 and later do not support space lines.
FCU reports code conversion table errors.	If you specified your own code conversion table, make sure that the file name and path are correct. FCU may also report code conversion table errors when the FX volume definition file contains both mainframe and OPEN-*x*FX volumes. Keep the FXoto volume definition file separate from the FXmto/otm volume definition file.

Error Codes and Messages

The error information returned by the **datasetGetLastError** function includes the FAL error information defined in the **dataset.h** file. Table 7-2 lists and describes the FAL error codes and provides instructions for resolving each error condition. In Table 7-2, the error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in Table 7-2, check for illegal I/O access contention for the FX volume between the z/OS and open-system hosts.

The FAL error logs for UNIX are **/tmp/fal_error** and **/tmp/fal_error.bak**, and **/tmp/fal_dump** and **/tmp/fal_dump.bak**. The FAL logs for Windows 2000/2003/Windows NT systems are **c:\fal_error** and **c:\fal_error.bak**, and **c:\fal_dump** and **c:\fal_dump.bak**.

Note: Error codes with a negative value are FAL errors. Error codes with a positive value are system errors. UNIX system error codes are defined in the standard error file **errno.h**.

Table 7-2 FAL Error Codes

Error Code	Error Message and Description	Recommended Action(s)
-7*	DATASET_ERROR_INVALID_VOLUME The actual VSN and the VSN specified in the FX volume definition file do not match.	Make sure that the VSN in the FX volume definition file is correct.
-8	DATASET_ERROR_DATASET_NOT_FOUND The target dataset was not found.	Make sure that the actual dataset name and the specified dataset name are the same. You can use the MF-File list command in the FCU HELP menu, or VTOC dump data on the z/OS host, to check the dataset name.
-9	DATASET_ERROR_NOT_SUPPORTED The data format is not supported.	Make sure that the dataset was created correctly on the z/OS host.
-10*	DATASET_ERROR_DEVICE_TYPE_NOT_SUPPORTED The device emulation type is not supported.	Make sure that the device emulation type (LVI) is correct in the FX volume definition file. The supported LVIs are 3390-3A, -3B and -3C.
-11	DATASET_ERROR_DSORG_NOT_SUPPORTED The dataset organization type is not supported.	Check the DO type using the MF-File list command in the FCU HELP menu, or VTOC dump data on the z/OS host.
-12	DATASET_ERROR_RECFM_NOT_SUPPORTED The record format is not supported.	Check the RF type using the MF-File list command in the FCU HELP menu, or VTOC dump data on the z/OS host.
-13*	DATASET_ERROR_INVALID_DATA The data in the VTOC or the dataset is invalid.	Make sure that the VTOC and dataset were created correctly on the z/OS host.
-14*	DATASET_ERROR_VOLUME_DEFINITION_INVALID The format of volume definition file is invalid.	Make sure that the FX volume definition file was created correctly.
-15	DATASET_ERROR_DATASET_NOT_OPENED An attempt was made to read the dataset without opening it.	Make sure that the **datasetOpen** function is called before the **datasetGet** function.

Error Code	Error Message and Description	Recommended Action(s)
-16	DATASET_ERROR_DATASET_NOT_CLOSED An attempt was made to open the dataset without closing it first.	Make sure the requirements and restrictions specified in Chapter 6 are met. ***For example:*** Dataset open and close must be used as a pair. More than one dataset cannot be open within one process. datasetOpen, datasetGetFileInformation, and datasetFindFirstFile cannot be used while the dataset is being accessed by datasetGetFileInformation or datasetFindFirstFile. datasetGetFileInformation and datasetFindFirstFile cannot be used while the dataset is being accessed datasetOpen.
-17	DATASET_ERROR_BUFLEN_SHORT The buffer length specified by **datasetGet** is shorter than the actual record length.	Make sure that the buffer area is larger than the dataset record length.
-18*	DATASET_ERROR_VOLUME_LABEL_INVALID No standard volume label was found, or the contents of the VTOC are illegal.	Make sure that volume initialization is complete and correct on the z/OS host. This error occurs when a system that does not support large files accesses a formatted volume from a system that supports large files. This error also occurs when a data partition size is incorrect for Solaris.
-19*	DATASET_ERROR_VTOC_INVALID No VTOC found, or contents of VTOC are invalid.	Make sure that the VTOC was created correctly on the z/OS host.
-20*	DATASET_ERROR_VOLUME_NOT_DEFINED The specified volume is not defined.	Make sure that the specified volume has been entered correctly in the FX volume definition file.
-21	DATASET_ERROR_INVALID_ARGUMENT An argument of the function is invalid.	Make sure that the argument for the FAL function is correct.
-22	DATASET_ERROR_NO_DATASET No dataset was found.	Make sure that the dataset has been created correctly on the z/OS host.
-23*	DATASET_ERROR_NON_STANDARD_R0_EXIST Nonstandard record 0 (R0) exists.	Change the R0 track format to standard track format. FAL cannot write on tracks with nonstandard R0.
-24	DATASET_ERROR_INVALID_MODE The **mode** argument of **datasetOpen** is not valid.	Make sure that the value of the **mode** argument for the **datasetOpen** function is either **r** (for read) or **w** (for write).
-25*	DATASET_ERROR_VOLUME_DEFINED_READ_ONLY The open-system host tried to write to a read-only volume.	Make sure that the target dataset for an open-system write operation is on a 3390-3A/C volume.
-26	FAL_INTERNAL_ERROR Internal error of FAL	Collect error logs file and error dump file. And make contact a maintenance staff. This error occurs when the open system does not have enough memory.
-27*	DATASET_ERROR_END_OF_VOLUME The end of volume was detected before the end of dataset was detected.	The open-system volume/partition size is smaller than the z/OS volume size. Make sure that the partition size is specified correctly on the open system. This error occurs when the open system disk is full or it exceeds a limitation for FXmto.
-28	DATASET_ERROR_OVERFLOW Data cannot be written because the dataset is full.	Check the size of the data to be written, and extend the size of the dataset as needed.

Error Code	Error Message and Description	Recommended Action(s)
-33	DATASET_ERROR_PARAMETER_MISMATCH User-specified RF, BL, RL does not match the RF, BL, RL defined in the VTOC; or RF, BL, RL not specified and not defined in VTOC.	Make sure to specify the correct VSE record option parameters when accessing VSE datasets (see section Record Description Word (RDW) Option).
-35	DATASET_ERROR_NO_LICENSE FAL can't permit execution of software that doesn't have a software license.	Ensure that the software license is current and correct. If problems persist, please contact the Hitachi Data Systems Support Center..
-36	DATASET_ERROR_TIMEOUT_LICENSE FAL can't permit execution of software with an expired software license trail time.	Ensure that the trial software license is current and correct. If problems persist, please contact the Hitachi Data Systems Support Center
-37	DATASET_ERROR_HOSTNAME_CHANGE FAL can't permit execution if the current host and the installed host are not identical and/or the hostname is changed.	Ensure that the current host name has not been changed.
-39	DATASET_ERROR_MULTI_VOLUME_DEFINITION_ RECORD_OVER The number of parameter sets for multiple volume definition file exceeded 1000.	Parameter sets more than 1000 cannot be processed. Decrease them not to exceed 1000.
-40	DATASET_ERROR_MULTI_VOLUME_NO_DATASET The data set isn't exist in the next volume.	Check volume serial number in the multiple volume definition file.
-41	DATASET_ERROR_MULTI_VOLUME_NO_TRANSFER	Data cannot be transferred to the dataset that is in middle volume of multiple volume.
-42	DATASET_ERROR_MULTI_VOLUME_DEFINITION_ INVALID_RECORD_LENGTH The record length in the multiple volume definition file is too long.	Specify the record length less than 1400 characters.(not include delimiter).
-43	DATASET_ERROR_MULTI_VOLUME_DEFINITION_ PARAMETER_ERROR The number of volume for one dataset in the multi volume definition file exceeded 31.	Specify the number of volume is less than 31 for one line in the multi volume definition file.
-44	DATASET_ERROR_MULTI_VOLUME_DEFINITION_ NO_DATASET The Dataset name is not specified in the multiple volume definition file.	Specify the dataset name in the head volume information of the multiple volume definition file.
-45	DATASET_ERROR_MULTI_VOLUME_DEFINITION_VSN_ LENGTH_ERROR VSN is incorrect in the multiple volume definition file.	Check if VSN length in the multiple definition file is less than 7.
-46	DATASET_ERROR_MULTI_VOLUME_DEFINITION_DSN_ LENGTH_ERROR DSN is incorrect in the multiple volume definition file.	Check if DSN length in the multiple definition file is less than 45.
-47	DATASET_ERROR_MULTI_VOLUME_DEFINITION_VOLID _LENGTH_ERROR The VSN identification length in the multiple volume definition file is too long.	Specify the VSN identification length less than 36 characters.
-48	DATASET_ERROR_MULTI_VOLUME_DEFINITION_ NO_NEXT_VOLUME The next VSN is specified in the multiple volume definition file when the volume isn't last on VTOC.	Specified all volumes in the multiple volume definition file.

Error Code	Error Message and Description	Recommended Action(s)
-50*	DATASET_ERROR_END_OF_FILE End of File (EOF) was detected.	None.
-51*	DATASET_ERROR_END_OF_VTOC End of VTOC was detected.	None.

FCU Error Codes for UNIX

If FCU for UNIX reports an error, use the **Help-Error** command to view the most recent error. Table 7-3 lists and describes the FCU error codes for UNIX and provides instructions for resolving each error condition. In Table 7-3, the error codes marked by an asterisk (*) may also be reported when I/O access contention for the FX volume occurs between the z/OS and open-system hosts. If the cause of the error cannot be identified as described in Table 7-3, check for any illegal I/O contention for the FX volume.

Note: Error codes with a negative value are FCU errors. Error codes with a positive value are system errors. UNIX system error codes are defined in the standard error file **errno.h**.

Note: The error codes with "(C)" in the "Error code" cell in Table 7-3 are generated only when using with FX Code Converter. For details, please see the *Hitachi Lightning 9900 V Series FX Code Converter User's Guide* (MK-93RD152).

Table 7-3 FCU Error Codes for UNIX

Error Code	Error Message and Description	Recommended Action(s)
-100	No parameter file The FCU parameter definition file could not be found.	If you specified the parameter definition file using the [**param**] option, make sure that the specified file exists and the name is correct. If you did not specify the [param] option when you started FCU, make sure that the default parameter definition file exists (fcudata.param in the current directory).
-101*	Parameter file: Open error An error occurred when opening the parameter definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-102*	Parameter file: Read error An error occurred when reading the parameter definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-103*	Parameter file: No valid data The parameters in the parameter definition file are not valid.	Make sure that the FCU initiation parameters are entered correctly in the parameter definition file.
-107	Parameter file: CODE_CONV error The code conversion specified in the parameter definition file is not valid.	Make sure that the code conversion is specified as either **EA** or **No**.
-108	Parameter file: PADDING error The padding option specified in the parameter definition file is not valid.	Make sure that the padding is specified as either **Yes** or **No**.

Error Code	Error Message and Description	Recommended Action(s)
-109	Parameter file: DELIMITER error The delimiter option specified in the parameter definition file is not valid.	Make sure that the delimiter is specified as either **CR**, **LF**, or **No**.
-110*	Parameter file: Open error An error occurred when opening and outputting the parameter definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-111	Parameter file: Write error An error occurred when writing to the parameter definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-112	Parameter file: Close error An error occurred when closing the parameter definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-114	Parameter: No input file name The input file name was not specified.	Make sure to specify the input file name.
-115	Parameter: VSN error The specified VSN is not correct.	Make sure that the specified VSN matches the actual VSN. Make sure that the VSN is separated from the dataset name by a colon (:).
-116	Parameter: Input file name error The specified input file name is not correct.	Make sure that the specified file name matches the actual file name.
-117	Parameter: Dataset name error The specified input dataset name is not correct.	Make sure that the specified dataset name matches the actual dataset name.
-118	Parameter: Output file name error The specified output file name is incorrect.	Make sure that the specified output file name matches the actual output file name.
-119*	Input file: Open error An error occurred when acquiring the dataset attribute information of the input file.	Display the error code using the **Help-Error** command. If an FAL error code is displayed, refer to Table C.1. If a system error code is displayed, please refer to the OS user manual. For example, if a partition name does not match the partition name in the volume definition file, system error code 6 (No such device) is displayed.
-120	Overwrite ? (OK/Cancel) This message asks you to confirm whether to overwrite the existing file.	The specified open-system target file already exists. Select **OK** to overwrite the file, or select **Cancel** to specify a different target file.
-121	Output file: File name error The output file name is not specified.	Make sure that the correct output file name is specified.
-122*	Output file: Open error An OPEN error occurred when checking to see if the output file exists.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.

Error Code	Error Message and Description	Recommended Action(s)
-124	Volume definition: MFtype error. Incorrect MFtype is specified in the volume definition file.	Specified MFN or MFA in MFtype of the volume definition file.
-125*	Volume definition: VSN error The VSN specified in the volume definition file is incorrect.	Display the contents of the volume definition file using the **Help-Volume** command. Make sure that the VSN for the specified volume is correct.
-126	Volume definition: Partition name error The partition name specified in the volume definition file is incorrect.	Display the contents of the volume definition file using the **Help-Volume** command. Make sure that the partition name is correct.
-127*	Volume definition: Emulation type error The LVI type specified in the volume definition file is incorrect.	Display the contents of the volume definition file using the **Help-Volume** command. Make sure that the LVI type is correct.
-128*	Volume definition file: Open error An error occurred when opening the volume definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors. For example, if the volume definition file does not yet exist, error code 2 (No such file or directory) is displayed.
-129*	Volume definition file: Read error An error occurred when reading the volume definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-130*	Volume definition file: No data The information found in the volume definition file is not valid.	Display the contents of the volume definition file using the **Help-Volume** command. Make sure that the parameters for each volume are correct.
-131	Volume definition file: Close error An error occurred when closing the volume definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-135	Parameter error: No input file name The input VSN is not specified.	Specify the VSN of the z/OS source dataset before selecting the **Help-MF-File** command.
-136	Parameter error: VSN error The input VSN is incorrect.	Make sure that the VSN has six characters.
-137	Dataset error: No dataset The specified volume has no datasets.	Make sure that the VSN is correct.
-138*	Dataset error: Search error An error occurred when searching the dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to Table C.1. If a system error code is displayed, please refer to the OS user manual.
-139	Dataset error: Close error An error occurred when closing the dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to Table C.1. If a system error code is displayed, please refer to the OS user manual.
-140	Input file error: Invalid organization type The DO type of the dataset is not supported.	Display the attribute information using the **Help-MF-File** command. The DO type must be SAM.

Error Code	Error Message and Description	Recommended Action(s)
-141	Input file error: Invalid record format The RF type of the dataset is not supported.	Display the attribute information using the **Help-MF-File** command. The RF type must be fixed-length or variable-length.
-142	Input file error: Invalid block length The block length of the dataset is invalid.	Display the attribute information using the **Help-MF-File** command. The block length must be nonzero and cannot be greater than 32 kB.
-143	Input file error: Invalid record length The record length of the dataset is invalid.	Display the attribute information using the **Help-MF-File** command. The record length must be nonzero and cannot be greater than 32 kB.
-144*	Input file error: No data No data was found in the specified dataset.	Display the attribute information using the **Help-MF-File** command, and check the dataset size.
-150*	Input file: Open error A file open error occurred in the input dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to Table C.1. If a system error code is displayed, please refer to the OS user manual.
-151*	Output file: Open error A file open error occurred in the output UNIX file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-152	Output file: Get file data error A data acquisition error of the output file occurred during an FXotm operation.	Collect information such as error log for troubleshooting.
-153	Processing data: Length check error A data length to be processed by FXotm does not match.	Make sure that the specified data length matches the actual data length. Collect information such as error log for troubleshooting.
-155	Buffer: Memory allocation error Memory allocation failed.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-160*	Input file: Read error A read error occurred in the input dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to Table C.1. If a system error code is displayed, please refer to the OS user manual.
-161*	Output file: Write error A write error occurred in the output UNIX file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-162	Output file: Code conversion error An error occurred in the code conversion to the output UNIX file.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to Table C.1.
-163	Get processing data error The acquisition of processing data failed.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to Table C.1. If a system error code is displayed, please refer to the OS user manual.
-165	Dataset error: Invalid data An invalid record length was found in the dataset.	Make sure that the z/OS dataset was generated correctly.
-170	Input file: Close error A file close error occurred in the input dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to Table C.1. If a system error code is displayed, please refer to the OS user manual.

Error Code	Error Message and Description	Recommended Action(s)
-171	Output file: Close error A file close error occurred in the output UNIX file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-180	UNIX/Open system file: Invalid directory name The specified directory name is not valid.	Check the specified directory name.
-181	UNIX file: Not a directory The specified name is not a directory name.	Check the specified directory name.
-182*	UNIX/Open system file: Open directory error A directory open error occurred.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-183	UNIX/Open system file: Close directory error A directory close error occurred.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-190	Input file name: No data The input file name is not specified.	The input file name must be specified when you select **File-Save**.
-191	Output file name: No data The output file name is not specified.	The output file name must be specified when you select **File-Save**.
-192	Parameter file name: No data The parameter definition file name is not specified.	The FCU parameter definition file name must be specified when you select **File-Save**.
-200	Parameter file: End line The last parameter set was loaded from the parameter definition file.	The next time you select **File-Load**, the first set of parameters will be loaded.
-201	Parameter file: Direction error The data transfer direction specified in the parameter file is incorrect.	Make sure that the direction (**mto** or **otm**) is correct.
-202	Parameter file: Too many data The number of parameter sets for parameter definition file exceeded 100.	The FCU parameter definition file can only store a maximum of 100 parameter sets. If necessary, delete one or more parameter sets to make room for a new parameter set.
-203	Parameter: Empty select error The **Emp** parameter is incorrect.	Make sure that the **Emp=Yes/No** parameter is correct.
-204	Parameter: RDW select error The **RDW** parameter is incorrect.	Make sure that the **RDW=Yes/No** parameter is correct.
-205	RDW error: CODE_CONV not supported Code conversion is not specified as **No** when **RDW=Yes**.	Code conversion cannot be performed when **RDW=Yes**. Change the code conversion parameter to **No**.

Error Code	Error Message and Description	Recommended Action(s)
-206	RDW error: PADDING not supported Padding is not specified as **No** when **RDW=Yes**.	Padding cannot be processed when **RDW=Yes**. Change the padding parameter to **No**.
-207	RDW error: DELIMITER not supported Delimiter is not specified as **No** when **RDW=Yes**.	Delimiters cannot be processed when **RDW=Yes**. Change the delimiter parameter to **No**.
-210	Parameter file: Comment line This is a comment line in the parameter file.	If you specify **Load**, FCU will move to the next line. You can also replace the comment line with a valid parameter.
-220	Parameter: VSE select error The VSE parameter format is not correct.	Make sure that the number of VSE parameters is correct and that a comma is used correctly to separate the VSE parameters.
-221	Parameter: VSE record format error Record format in the VSE parameter is not correct.	Make sure that the record format is set to either one of F/FB/V/VB.
-222	Parameter: VSE record length error Record length in the VSE parameter is not correct.	Make sure that the record length is set to the correct value within the extent allowed.
-223	Parameter: VSE block length error Block length in the VSE parameter is not correct.	Make sure that the block length is set to the correct value within the extent allowed.
-230	No code conv. table file: No code conv. table The code conversion table was not found.	Make sure that the code conversion table file name is correct and that the file exists. This error may also be reported if you mix mainframe and OPEN-x devices in the same FX volume definition file.
-231	Code conv. table: Open error The code conversion table could not be opened.	Refer to the OS user manuals for assistance.
-233	Code conv. table: Close error The code conversion table could not be closed.	Refer to the OS user manuals for assistance.
-234	Code conv. table: Get file data error The size of the code conversion table could not be obtained.	Check the contents of the file. Refer to the OS user manuals for assistance.
-235	Code conv. table: File size error The size of the code conv table is not correct.	Make sure that the size of the code conversion table is 256 bytes.
-236	Code conv. table function: Invalid argument No source data to be converted was found.	Check the contents of the input file, especially the delimiters.
-238	Code conv. table name: No data The file name of the code conversion table is not specified.	If you do not specify **EA** or **No** for the code conversion option, make sure to specify the correct file name of your code conversion table.

Error Code	Error Message and Description	Recommended Action(s)
-300	Data error: Invalid record length The data length is not correct for the FXotm padding function.	Check the source data length and the target record length, and make sure that the record length is correct for the source data entities.
-301	Dataset error: Invalid record format The record format is not correct for the FXotm padding function.	For FXotm with padding, make sure that the target dataset has fixed-length record format.
-302	Parameter error: Delimiter error The delimiter setting is not correct for the FXotm padding function.	If padding=Yes for an FXotm operation, the delimiter option must be CR, LF or CRLF.
-319*	Dataset: Open error An error occurred when opening the dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to Table C.1. If a system error code is displayed, please refer to the OS user manual. For example, if the partition name does not match the partition name in the volume definition file, system error code 6 (No such device) is displayed.
-324	O to M error: RDW is not supported	Do not specify the RDW option for FXotm operations.
-340	Dataset error: Invalid organization type The DO type of the dataset is not supported.	Display the attribute information using the **Help-MF-File** command. The DO type must be SAM.
-341	Dataset error: Invalid record format The RF of the dataset is not supported.	Display the attribute information using the **Help-MF-File** command. The RF type must be fixed-length or variable-length.
-342	Dataset error: Invalid block length The block length of the dataset is invalid.	Display the attribute information using the **Help-MF-File** command. The block length must be nonzero and cannot be greater than 32 kB.
-343	Input file error: Invalid record length The record length of the dataset is invalid.	Display the attribute information using the **Help-MF-File** command. The record length must be nonzero and cannot be greater than 32 kB.
-350*	Input file: Open error An open error occurred in the input UNIX file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-351*	Output file: Open error A file open error occurred in the output dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to Table C.1. If a system error code is displayed, please refer to the OS user manual.
-352	Input file: Get file data error A data acquisition error for input file occurred during an FXotm operation.	Collect information such as error log for trouble shooting.
-353	Processing data: Length check error A data length to be processed in FXotm operation does not match.	Collect information such as error log for trouble shooting.
-355	Buffer: Memory allocation error Memory allocation failed.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.

Error Code	Error Message and Description	Recommended Action(s)
-360*	Input file: Read error A read error occurred in the input UNIX file.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to Table C.1. If a system error code is displayed, please refer to the OS user manual.
-361*	Output file: Write error A write error occurred in the output dataset.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-362	Output file: Code conversion error An error occurred in the code conversion to the output dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to Table C.1.
-363	Get processing data error The acquisition of processing data failed.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to Table C.1. If a system error code is displayed, please refer to the OS user manual.
-370	Input file: Close error A file close error occurred in the input UNIX file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-371	Output file: Close error A file close error occurred in the output dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to Table C.1. If a system error code is displayed, please refer to the OS user manual.
-379*	UNIX file: No data No data was found in the input UNIX file.	Make sure to specify an input file which contains data.
-380	No UNIX file The specified UNIX file was not found.	Make sure that the specified UNIX file exists.
-381*	UNIX file: Open error An open error occurred in the UNIX file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-382	Output file: Unsupported record format The record format of the output file is not supported.	Display the attribute information using the **Help-MF-File** command. The RF type must be fixed-length or variable-length.
-383*	Input file: Invalid format The format of the input file is incorrect.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-384*	Input file: Invalid delimiter position The delimiter position in the input file is incorrect. Data record length of input file exceeds that of target dataset, or a record with no data entity is included.	Display the attribute information using the **Help-MF-File** command. Make sure that the record length of the target dataset is correct.
-385	Input file: File seeking error An error occurred when seeking for the input file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.

Error Code	Error Message and Description	Recommended Action(s)
-399	Volume definition : VSN identification length error. The VSN identification length in the volume definition file is too long.	Specify the VSN identification length than 35 characters.
-400	Parameter: Invalid input file name More than one input file name was specified.	Specify only one file name as the input file.

FCU Error Codes for Windows Systems

If FCU for Windows 2000/2003/Windows NT systems reports an error, use the **View-Error information...** command to view the most recent error. FCU for Windows 2000/2003/Windows NT systems also logs errors in the FCU log file (e.g., **fcudata.prm.log**). Table 7-4 lists and describes the FCU error codes for Windows 2000/Windows NT systems and provides instructions for resolving each error condition.

Note: Error codes with a plus value are system errors. Windows 2000/2003/Windows NT systems system error codes are defined in the **errno.h** file attached with Microsoft Visual C++®.

Table 7-4 FCU Error Codes for Windows 2000/2003/Windows NT Systems

Error Code	Error Message and Description	Recommended Action(s)
-100	Parameter definition file: Open error An error occurred when opening the parameter definition file.	Make sure that the parameter definition file was created correctly. If the parameter definition file was created correctly, check the system error.
-101	Parameter: Count error An error is detected in the parameter count.	Make sure that the parameter count is correct.
-102	Parameter: Direction error The data transfer direction is not correct.	Make sure that the direction is specified correctly as **mto** or **otm**.
-103	Parameter: Mainframe file name error Mainframe file name is not correct.	Make sure that the mainframe file name is set correctly.
-104	Parameter: Open system file name error Open system file name is not correct.	Make sure that the open system file name is set correctly.
-105	Parameter: Code conversion error Code conversion setting is not correct.	Make sure that the code conversion option is specified as **EA**, **EcA**, **No**, or **File_name** (of your code conversion table). This error may also be reported if you mix 3390/3380 and OPEN-x devices in the same FX volume definition file.
-106	Parameter: Padding error Padding setting is not correct.	Make sure that the padding option is specified as **Yes** or **No**.
-107	Parameter: Delimiter error Delimiter setting is not correct.	Make sure that the delimiter option is specified as **CRLF** or **No**.
-108	Parameter: Add parameter error Delimiter setting is not correct.	If you are adding delimiters for Windows 2000/2003/Windows NT systems, make sure that the delimiter option is specified as **CRLF** (not just **CR** or **LF**).
-109	Parameter: Empty duplication error More than one empty setting is specified.	Specify only one empty setting.
-110	Parameter: RDW duplication error More than one RDW setting is specified.	Specify only one RDW setting.
-120	Volume definition file: Open error An error is detected when opening the volume definition file.	Make sure that the volume definition file was created correctly. If the volume definition file is correct, check the system error.

-121	Volume definition file : Length error The record length in the volume definition file is too long.	Specify the record length less than 2080 characters (not including delimiter).
-124	Volume definition: Emulation type Length error An emulation type parameter in the volume definition file is too long.	Specify an emulation type parameter less than 11 characters.
-125	Volume definition: MFtype Length error. Incorrect Ftype is specified in the volume definition file.	Specified MFN or MFA in MFtype of the volume definition file.
-126	Volume definition: VSN identification length error. The VSN identification length in the volume definition file is too long.	Specify the VSN identification length than 35 characters.
-130	Dataset: No dataset error No dataset is found.	Make sure that the mainframe name is specified correctly, or that the dataset is allocated correctly on the specified volume.
-131	Dataset: Search error An error is detected in searching the dataset.	Make sure that the volume definition file name is specified correctly, or that the mainframe file name is specified correctly.
-132	Dataset: Information get error An error is detected in acquiring dataset information.	Make sure that the volume definition file name is specified correctly, or that the mainframe file name is specified correctly.
-133	Dataset: Organization error The specified dataset org. type is not correct.	Make sure that the dataset organization type is specified correctly.
-134	Dataset: Record format error The specified record format is not correct.	Make sure that the record format is specified correctly.
-135	Dataset: Block length error The specified block length is not correct.	Make sure that the block length is specified correctly.
-136	Dataset: Record length error The specified record length is not correct.	Make sure that the record length is specified correctly.
-137	Dataset: Dataset size error The specified dataset size is not correct.	Make sure that the dataset size is specified correctly.
-138	Dataset: Close error An error is detected during close operation.	Check the FAL error code and system error code.
-150	Mainframe file: Open error An error is detected when opening the mainframe file.	Check the FAL error code and system error code.
-151	Mainframe file: Read error An error is detected during reading data from the mainframe file.	Check the FAL error code and system error code.
-152	Mainframe file: Write error An error is detected when writing data into the mainframe file.	Check the FAL error code and system error code.
-153	Mainframe file: Close error An error is detected when closing the mainframe file.	Check the FAL error code and system error code.

-154	Mainframe file: Record format error An error is detected in the record format of the mainframe file.	For FXotm with the padding function, make sure that the target dataset has fixed-length record format (or change padding to **No**).
-170	Open system file: Open error An error is detected when opening the open system file.	Make sure that the open-system file name is specified correctly. Check if any system error is reported.
-171	Open system file: Read error An error is detected when reading data from the open system file.	Check the system error.
-172	Open system file: Write error An error is detected when writing data into the open system file.	Check the system error.
-173	Open system file: Close error An error is detected when closing the open system file.	Check the system error.
-174	Open system file: No data error No dataset is found.	Make sure that the open-system file has data. If not, create the appropriate data in the open-system file.
-175	Open system file: Delimiter (CR) position error Delimiter (CR) position error is detected. The source data record length exceeds the target record length, or a record with no data entity is included.	Make sure that the open-system file name is correct. Make sure that the mainframe dataset name is correct. Make sure that the record length of the open-system file is correct.
-176	Open system file: Delimiter (LF) position error Delimiter (LF) position error is detected.	Make sure that the open-system file name is correct. Make sure that the mainframe dataset name is correct. Make sure that the record length of the open-system file is correct.
-177	Open system file: Record format error An illegal record format is found.	Make sure that the open-system file name is correct. Make sure that the mainframe dataset name is correct. Make sure the record format (fixed- or variable-length) of the open-system file data is correct.
-178	Open system file: Record length error An illegal record length was found. Data length of open-system file is too large.	Check the data length of the open-system file, and make sure the dataset has the correct record length.
-190	Code conversion error An error was found during code conversion.	Make sure that the dataset size is specified correctly.
-200	Process data get error An error is detected during close operation.	Check the FAL error code and system error code.
-220	External table file: Open error The code conversion table could not be opened.	Check the file name of code conversion table. Check the system error.
-221	External table file: Size error The code conversion table size is not correct.	Make sure that the size is 256 bytes and that the table was created correctly.
-222	External table file: Read error A read error was found when reading the code conversion table.	Check the system error.
-223	External table file: Close error The code conv. table could not be closed.	Check the system error.

-240	Parameter: Direction, PAD, and DEL not matched The combination of **otm** direction, **PAD=Yes**, and **DEL=No** is not allowed.	For FXotm with the padding function, make sure that the delimiter option is specified as **Yes** (or set padding=No).
-241	Parameter: Direction and RDW not matched The combination of **otm** data transfer direction and **RDW=Yes** is not allowed.	When the FX data transfer direction is **otm**, make sure that the RDW option is specified as **No**.
-242	Parameter: Code conv. and RDW not matched The combination of **RDW=Yes** and code conversion other than **No** is not allowed.	When the code conversion option is **EA** or **File_name**, make sure that the RDW option is specified as **No**. When RDW=Yes, the code conversion option must be specified as **No**.
-243	Parameter: Padding and RDW not matched The combination of **RDW=Yes** and **padding=Yes** is not allowed.	When the padding option is specified as **Yes**, make sure that the RDW option is specified as **No**. When the RDW option is specified as **Yes**, make sure that the padding option is specified as **No**.
-244	Parameter: Delimiter and RDW not matched The combination of **RDW=Yes** and **delimiter=Yes** is not allowed.	When the delimiter option is specified as **Yes**, make sure that the RDW option is specified as **No**. When the RDW option is specified as **Yes**, make sure that the delimiter option is specified as **No**.
-245	Parameter : Specified VOLSER isn't defined Volume Definition file. Specified VOLSER isn't defined the volume definition file.	Check whether specified VOLSER is defined in the volume definition file.
-300	Parameter definition file : Length error The record length in the parameter definition file is too long.	Specify the record length less than 3200 characters (do not include delimiter).
-301	Mainframe file name : Length error An input dataset name(in case of FXmto) or an output dataset name(in case of FXotm) in the parameter definition file is too long.	Specify an input/output dataset name less than 1025 characters.
-302	Opensystem file name : Length error An input filet name(in case of FXotm) or an output file name(in case of FXmto) in the parameter definition file is too long.	Specify an input/output file name less than 1025 characters.
-303	Code conversion Length error A code conversion file name in the parameter definition file is too long.	Specify a code conversion file name less than 1025 characters.
-304	VSE: Length error A VSE parameter in the parameter definition file is not corrected.	Specify a VSE parameter less than 21 characters.
-305	VSE record-format: Length error The record format for VSE in the parameter definition file is not corrected.	Specify the record format for VSE less than 3 characters.
-306	VSE record-length: Length error The record length for VSE in the parameter definition file is not corrected.	Specify the record length for VSE less than 6 characters.

-307	VSE block-length: Length error The block length for VSE in the parameter definition file is not corrected.	Specify the block length for VSE less than 6 characters.

Calling the Support Center

If you need to call the Hitachi Data Systems Support Center, make sure to provide as much information about the problem as possible, including:

- The circumstances surrounding the error or failure.
- The exact content of any error messages displayed on the host system(s).
- The exact content of any error messages displayed by Storage Navigator.
- The Storage Navigator configuration information (use the FD Dump Tool).
- The service information messages (SIMs), including reference codes and severity levels, displayed by Storage Navigator.
- Error codes: FCU error code, FAL error code, SYS error code. Use the FCU GUI to check recent error information (Help-Error command for UNIX, View-Error information command for NT).
- FCU parameters: direction (mto or otm), input and output files, and FCU options (code conversion, padding, delimiter, empty file, RDW, VSE record).
- FX volume definition file: contents
- FCU parameter definition file (if used): contents
- Command line log (if possible).
- FAL error logs. The FAL logs for UNIX are /tmp/fal_error and /tmp/fal_error.bak, and /tmp/fal_dump and /tmp/fal_dump.bak. The FAL logs for Windows 2000/2003/Windows NT systems are c:\fal_error and c:\fal_error.bak, and c:\fal_dump and c:\fal_dump.bak.
- Windows 2000/2003/Windows NT systems only: FCU log file (e.g., fcudata.prm.log), and Dr. Watson's log file (e.g., c:\WINNT\DRWTSN32.LOG).
- Syslog: error information and other applicable contents

The Hitachi Data Systems customer support staff is available 24 hours/day, seven days a week. If you need technical support, please call:

- United States: (800) 446-0744
- Outside the United States: (858) 547-4526

A

EBCDIC-ASCII Code Conversion

This appendix provides a list of EBCDIC to ASCII conversion values.

Table A-1 lists the EBCDIC-ASCII code conversions performed by the default code conversion table which is provided with FCU).

Table A-1 Default FCU EBCDIC-ASCII Conversions

(continues on following page)

Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII
00	NUL	NUL	20	DS		40	SP	DS	60	-	ENQ
01	SOH	SOH	21	SOS	a	41			61	/	BEL
02	STX	STX	22	FS	b	42			62		
03	ETX	ETX	23		c	43		s	63		
04	PF		24	BYP	d	44		t	64		
05	HT	RLF	25	LF	SMM	45		u	65		
06	LC	f	26	ETB	IL	46		v	66		
07	DEL	“	27	ESC	CUI	47		w	67		
08	GE	p	28		h	48		x	68		
09	RLF		29		i	49		y	69		
0A	SMM		2A	SW		4A		N	6A		
0B	VT	VT	2B	CUI		4B	.	ACK	6B	,	
0C	FF	FF	2C			4C	<	DC4	6C	%	LF
0D	CR	CR	2D	ENQ	HT	4D	(		6D	_	~
0E	SO	SO	2E	ACK	LC	4E	+	CUI	6E	>	
0F	SI	SI	2F	BEL	DEL	4F	|	@	6F	?	SUB
10	DLE	DLE	30			50	&	ETB	70		
11	DC1	DC1	31		j	51		z	71		
12	DC2	DC2	32	SYN	BS	52			72		
13	DC3	DC3	33		l	53			73		]
14	TM		34	PN	m	54			74		
15		e	35	RS	n	55		[	75		
16	BS	GE	36	UC	o	56			76		{

Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII
17	IL	g	37	EOT	PF	57			77		A
18	CAN	CAN	38		q	58			78		B
19	EM	EM	39		r	59			79	‘	-
1A	CC	k	3A		^	5A	!	SOS	7A	:	
1B	CUI		3B	CU3		5B	$	BYP	7B	#	
1C	IFS	IFS	3C	DC4	TM	5C	*	SW	7C	@	SP
1D	IGS	IGS	3D	NAK		5D	)		7D	‘	ESC
1E	IRS	IRS	3E			5E	:	CU3	7E	=	NAK
1F	IUS	IUS	3F	SUB	CC	5F	~	=	7F	“	FS
80		C	A0		J	C0	{	#	E0	\	*
81	a	/	A1		V	C1	A		E1		
82	b		A2	s		C2	B		E2	S	
83	c		A3	t		C3	C		E3	T	
84	d		A4	u		C4	D		E4	U	
85	e		A5	v		C5	E		E5	V	
86	f		A6	w		C6	F		E6	W	
87	g		A7	x		C7	G		E7	X	
88	h		A8	y	‘	C8	H		E8	Y	
89	i		A9	z	:	C9	I		E9	Z	!
8A		D	AA		K	CA		Y	EA		4
8B		E	AB		L	CB		Z	EB		5
8C		F	AC		M	CC			EC		6
8D		G	AD	[	$	CD			ED		7
8E		H	AE		O	CE			EE		8
8F		I	AF		P	CF			EF		9
90			B0		Q	D0	}	‘	F0	0	
91	j		B1		R	D1	J		F1	1	
92	k	,	B2			D2	K	.	F2	2	SYN
93	l	%	B3			D3	L	<	F3	3	
94	m	_	B4			D4	M	(	F4	4	PN
95	n	>	B5			D5	N	+	F5	5	RS
96	o	?	B6			D6	O	|	F6	6	UC
97	p		B7			D7	P	&	F7	7	EOT
98	q		B8		\	D8	Q		F8	8	
99	r		B9			D9	R		F9	9	
9A	^	;	BA		S	DA			FA		
9B			BB		T	DB			FB		
9C			BC		U	DC		0	FC		

Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII
9D			BD	]	)	DD		1	FD		
9E			BE		W	DE		2	FE		
9F		}	BF		X	DF		3	FF		

Acronyms and Abbreviations

ASCII	American National Standard Code for Information Interchange
BL	block length
CKD	count key data
CR	carriage return
CVS	custom volume size
D	delimiter
DAM	direct-access method (not supported by FX)
DAT	digital audio tape
DO	dataset organization
DS	dataset size
EA	EBCDIC/ASCII (includes EBCDIC-to-ASCII and ASCII-to-EBCDIC)
EBCDIC	extended binary-coded decimal interchange code
Emp	empty file option
EOF	end of file
ESCON	Enterprise System Connection (IBM trademark for optical channels)
F	fixed-length and de-blocking (z/OS record format)
FAL	File Access Library
FB	fixed-length and blocking (z/OS record format)
FC	fibre channel
FCU	File Conversion Utility
FD	floppy disk
FWD	fast-wide differential
GUI	graphical user interface
HMBR	Hitachi Multiplatform Backup/Restore
FX	Hitachi Cross-OS File Exchange
ICKDSF	A Device Support Facilities (DSF) command used to perform media maintenance
kB	kilobytes
LDEV	logical device
LF	line feed
LSM	Logical Storage Manager
LU	logical unit
LUN	logical unit number
LVI	Logical volume image

MF	mainframe
mto	mainframe-to-open
MVS™	Multiple Virtual Storage
OPEN-*x*	standard LU type, e.g., OPEN-3, OPEN-9
OS	operating system
otm	open-to-mainframe
oto	open-to-open
PAM	partitioned access method (not supported by FX)
R0	record 0
RDW	record description word
RF	record format
RL	record length
SAM	sequential-access method, System Administration Manager (HP-UX)
SCSI	small computer system interface
SMIT	System Management Information Tool (IBM AIX)
V	variable-length and de-blocking (z/OS record format)
VB	variable-length and blocking (z/OS record format)
VIR	Virtual LVI/LUN
volser	volume serial number
VOS3	Virtual-Storage Operating System 3 (a Hitachi z/OS OS)
VSAM	Virtual Storage Access Method (not supported by FX)
VSE	Virtual Storage Extended
VSN	volume serial number
VTOC	volume table of contents
WinNT	Windows NT systems

Index

A
accessing
  FCU, 7-2
  raw devices, 1-1
  VSE® datasets, 7-5

C
commands
  deinstallation, 3-20
  file menu, 4-5
  help menu, 4-5
  HRX file transfer operations, 2-3
  Parameter Delete, 4-13, 4-18
  Parameter Load, 4-13, 4-18
  Parameter Save, 4-13, 4-18
  Parameter-Save-Insert, 5-24
  reserve, 2-31
  TIME_OUT_VALUE, 2-12
  View Status bar, 4-13
  View Toolbar, 4-13
concurrent access support, 3-10
configuration
  all-mainframe, 1-2
  all-open, 1-1
  multiplatform, 1-1
configuration illustration
  oto, 2-3

D
delimiter option, 2-14
device recognition, 3-10

E
EBCDIC ASCII code conversion, 2-9

Q
queue depth, 3-11

R
reserve, 2-31

S
SAM
  sequential access method, 3-33
Sizing
  partition, 3-11
Support Center, Hitachi Data Systems, 7-22
system requirements, 3-2

T
target code values, 2-8
Time_Out_Value, 2-12

V
volume types
  in general, **Error! Not a valid bookmark in entry on page** 2-4
  mto, 2-4
  otm, 2-4, 2-5

Z
z-index entry, 1

Hitachi Data Systems

Corporate Headquarters
750 Central Expressway
Santa Clara, California 95050-2627
U.S.A.
Phone: 1 408 970 1000
www.hds.com
info@hds.com

Asia Pacific and Americas
750 Central Expressway
Santa Clara, California 95050-2627
U.S.A.
Phone: 1 408 970 1000
info@hds.com

Europe Headquarters
Sefton Park
Stoke Poges
Buckinghamshire SL2 4HD
United Kingdom
Phone: + 44 (0)1753 618000
info.eu@hds.com

Hitachi Data Systems

MK-96RD647-01